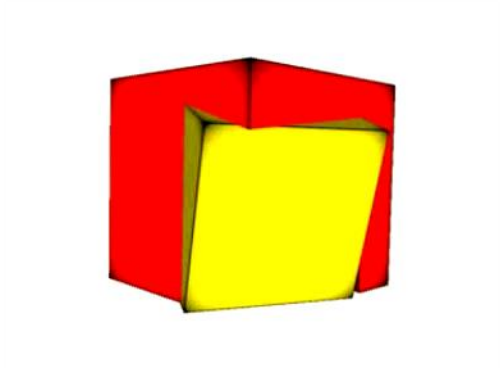
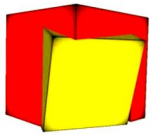
ТЕМА №16

Отсичане



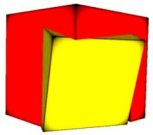


Съдържание

Тема 16: Отсичане

- Екранно отсичане
- Отсичане на отсечка
- Алгоритъм на Коен-Съдърленд
- Отсичане на многоъгълник

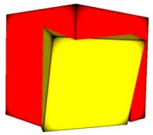
Екранно отсичане



Проблем

Проблем за решаване

- Растеризирането на примитиви е в рамките на правоъгълна зона
- Цял екран, прозорец или част от прозорец
- Няма нужда да се растеризира извън тази зона



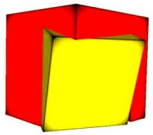
Цел на отсичането

Цел на отсичане (на англ. *clipping*)

- Да няма растеризиране извън видимата зона
- Да ускори растеризирането

Ускорение

- Не се растеризират целите примитиви, а само видимата им част
- Някои примитиви дори и не се рисуват



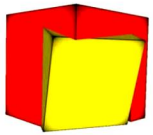
Особености

Отсичащи прави

- Почти винаги те са хоризонтални и вертикални: $x = const$ или $y = const$
- Отсичането е в правоъгълник

Координати

- Почти винаги те са цели числа
  (измерват разстояние в пиксели)



В тази лекция

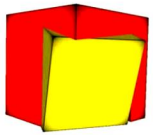
Ще се ограничим

- Правоъгълна зона на отсичане
- Ограничена от прави, успоредни на координатната система:

$$x_{min} \leq x \leq x_{max}$$
$$y_{min} \leq y \leq y_{max}$$

- Отсичане, приложимо и за векторна графика
- Отсичане на отсечки и многоъгълници

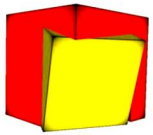
Отсичане на отсечка



Алгоритми за отсичане

Много алгоритми за отсичане на отсечки:

- Наивен алгоритъм (даже са два броя)
- На Коен-Съдърленд (Cohen-Sutherland)
- На Лианг-Барски (Liang-Barsky)
- На Никол-Лий-Никол (Nicholl-Lee-Nicholl)
- На Сайръс-Бек (Cyrus-Beck)



Наивни алгоритми

Първи наивен алгоритъм

- Растеризира се нормално
- Всеки пиксел се проверява за видимост

Особености

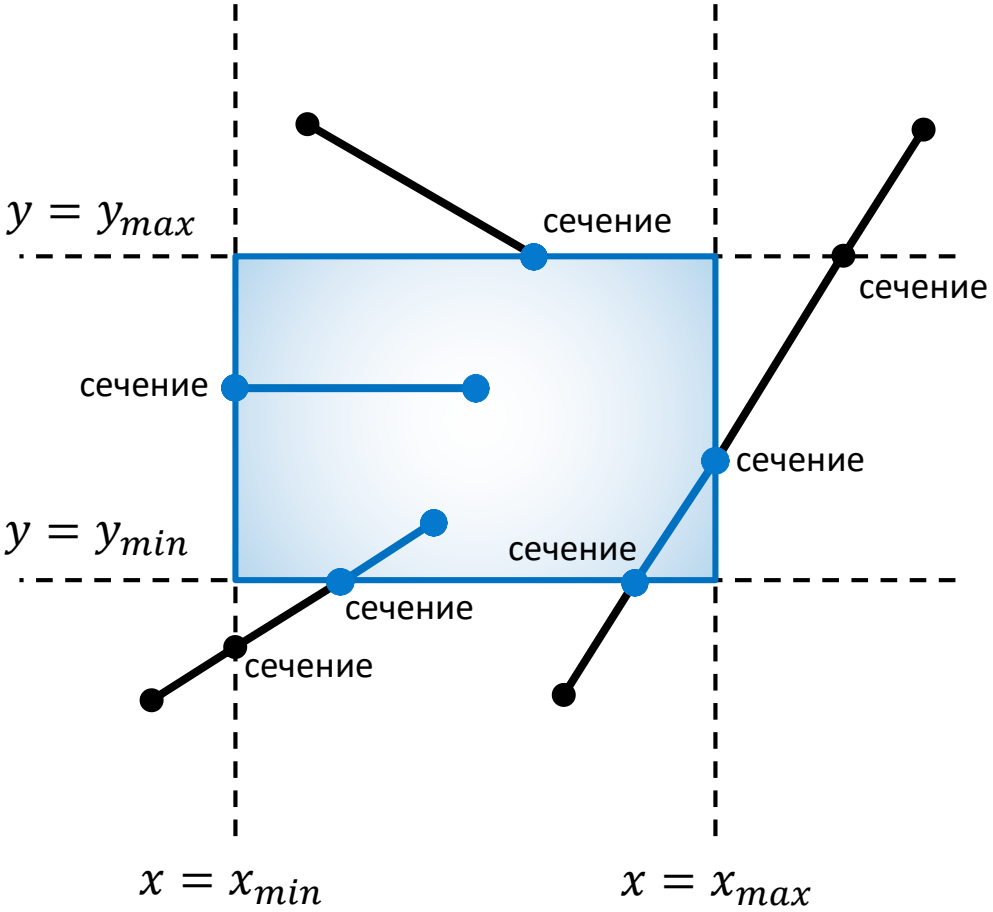
- Проверката за един пиксел е бърза
- Работи за всички примитиви
- Мноооого излишни изчисления

Втори наивен алгоритъм

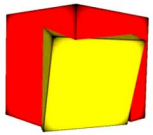
- Намират се сеченията на отсечките с ограничителните линии
- Премахват се точките, които са външни
- Ако остават две точки – те са краища на видимата отсечка. Иначе е или изцяло видима, или изцяло невидима

Особености

- Не зависи от дължината на отсечката
- Все пак прави излишни изчисления


$y = y_{max}$
$y = y_{min}$
$x = x_{min}$
$x = x_{max}$
сечение
сечение
сечение
сечение
сечение
сечение

Алгоритъм на Коен-Съдърленд



Обща идея

Обща идея

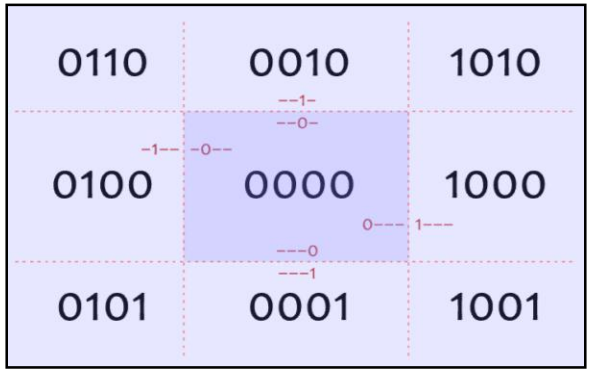
- Разделяме равнината на 9 области
- Всеки край на отсечка получава код според това в коя област се намира
- По двата кода разбираме дали отсечката е вътрешна, външна или пресичаща видимата зона
- Ако е пресичаща, кодът указва как да я скъсим и пак да я проверим

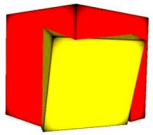
Кодиране на областите

- Използваме 4-битова маска
- Девет области с уникален код
- Можем сами да си избираме битовете

Примерно ето така

- Бит 0 = 1, ако $y < y_{min}$
- Бит 1 = 1, ако $y > y_{max}$
- Бит 2 = 1, ако $x < x_{min}$
- Бит 3 = 1, ако $x > x_{max}$





Свойства на кода

Ако краищата на отсечка са P и Q

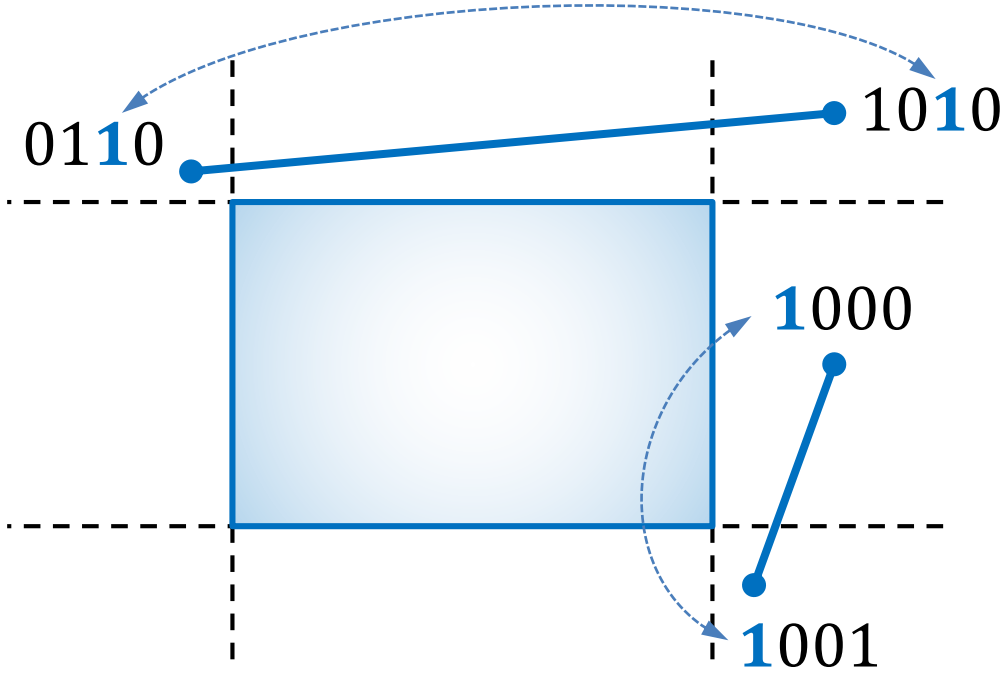
- Избираме кодовете им да са p и q

Бързи проверки с битови операции

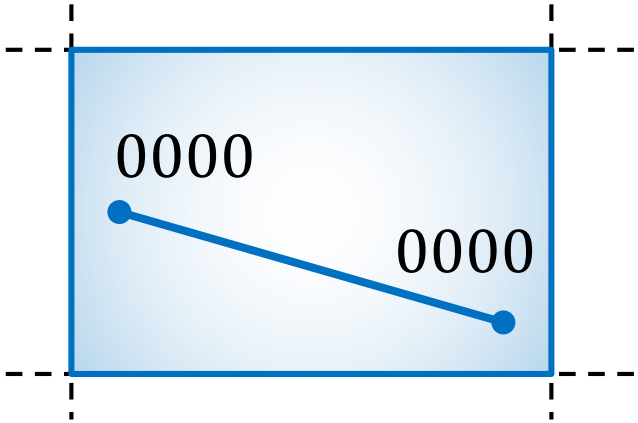
- Ако $p|q = 0$ отсечката е вътрешна
  (т.е. $p = 0$ и $q = 0 \Rightarrow P$ и Q са във видимата зона $\Rightarrow$ цялата отсечка е видима)
- Ако $p\&q \neq 0$ отсечката е външна
  (т.е. p и q имат общ ненулев бит $\Rightarrow P$ и Q са едновременно отвъд някоя права $\Rightarrow$ цялата отсечка е невидима)

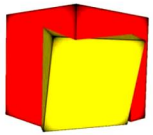
Примери

– Изцяло външна отсечка – двата ѝ кода имат поне по един общ ненулев бит



– Изцяло вписана отсечка – и двата кода на краищата ѝ са 0000





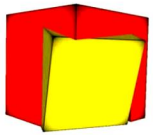
После какво?

Ако краищата имат ненулеви кодове, които са без общи ненулеви битове

- Отсечката потенциално пресича видимата зона
- Разделяме я на две части, такива, че едната да е изцяло външна (т.е. отпада от раз), а за другата си проверяваме кодовете

Как правим това?

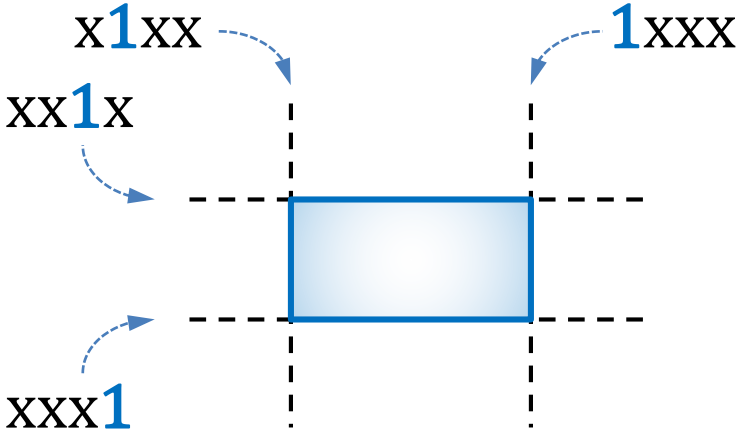
- Единият край има ненулев код (защо? 1т бонус)
- Харесваме си ненулев бит и изрязваме спрямо него
- Така получаваме две отсечки
- Едната забравяме веднага (външна е)
- Другата тестваме пак с $p|q = 0$ и $p\&q \neq 0$
- Така на няколко, но краен брой, стъпки отсичаме отсечката до видимата зона



Изрязване спрямо бит

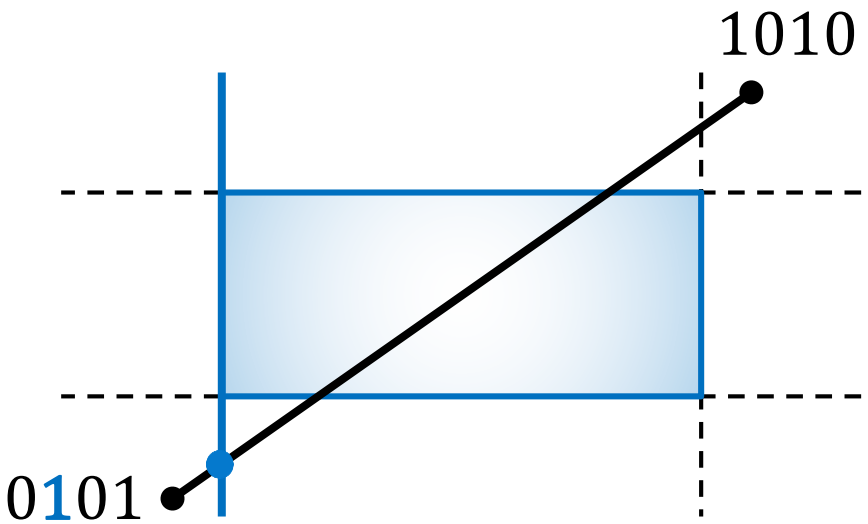
Определяне на правата за отсичане

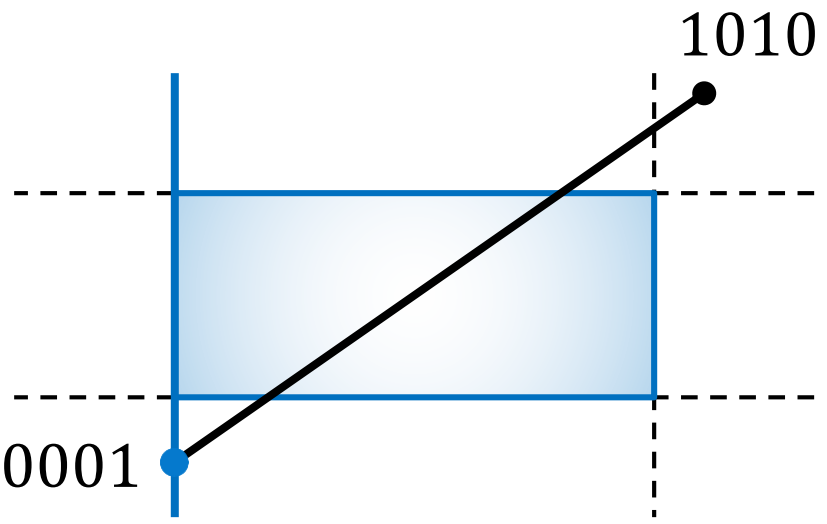
- Всеки бит е за зона **извън** дадена права
- По тази права отсичаме



Примери за отсичане

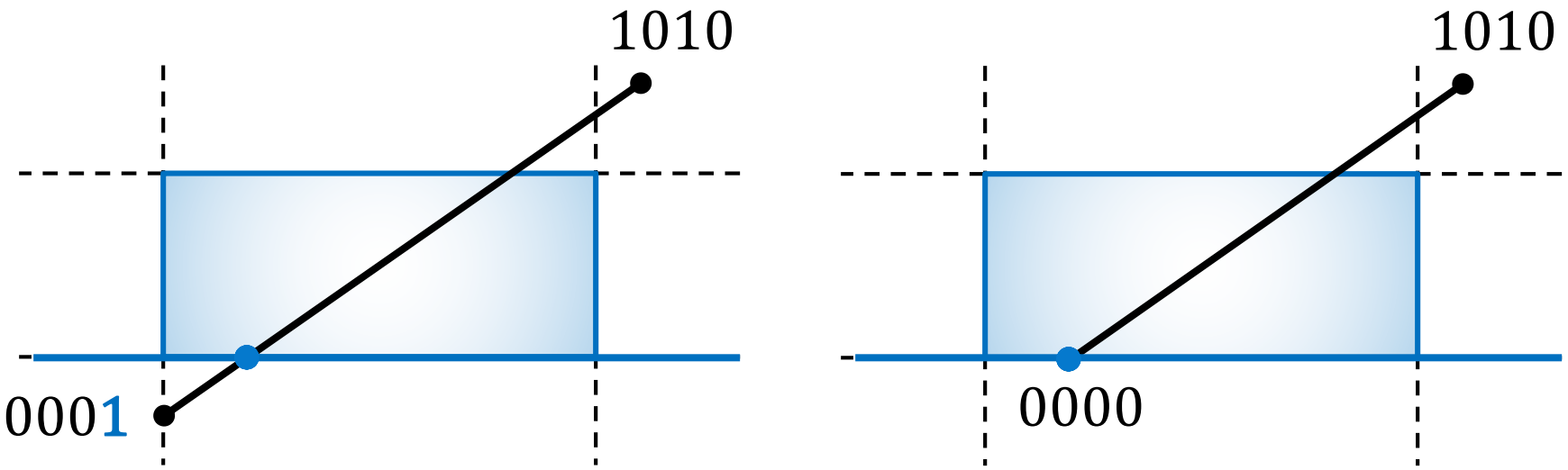
– По избран бит отсичаме по съответната му права





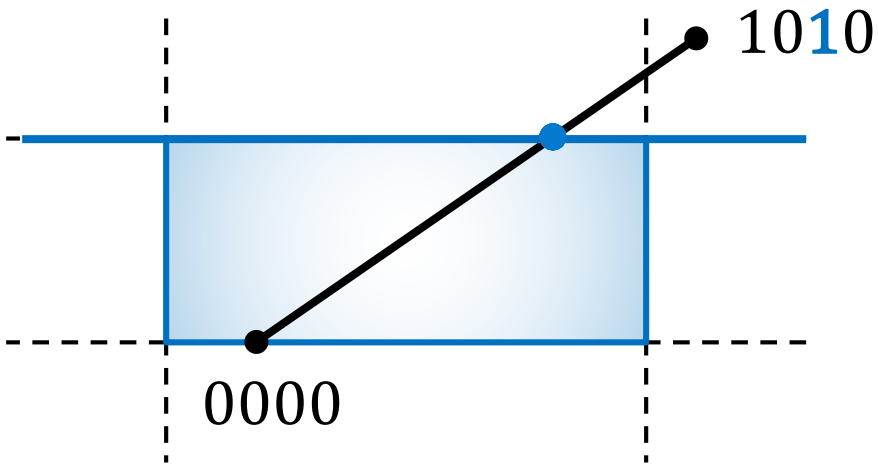
Продължаваме

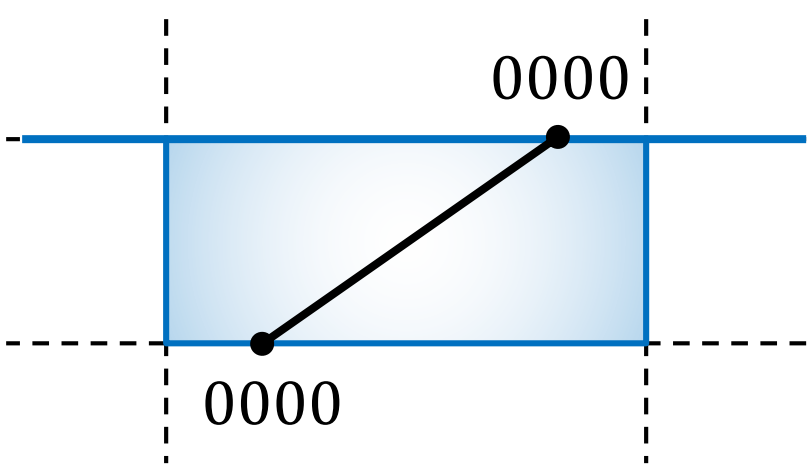
– Отсичаме по следващия бит

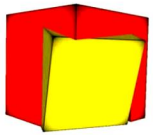


Продължаваме

- Сега и от другия край
- До получаване на $p|q = 0$ или $p\&q \neq 0$



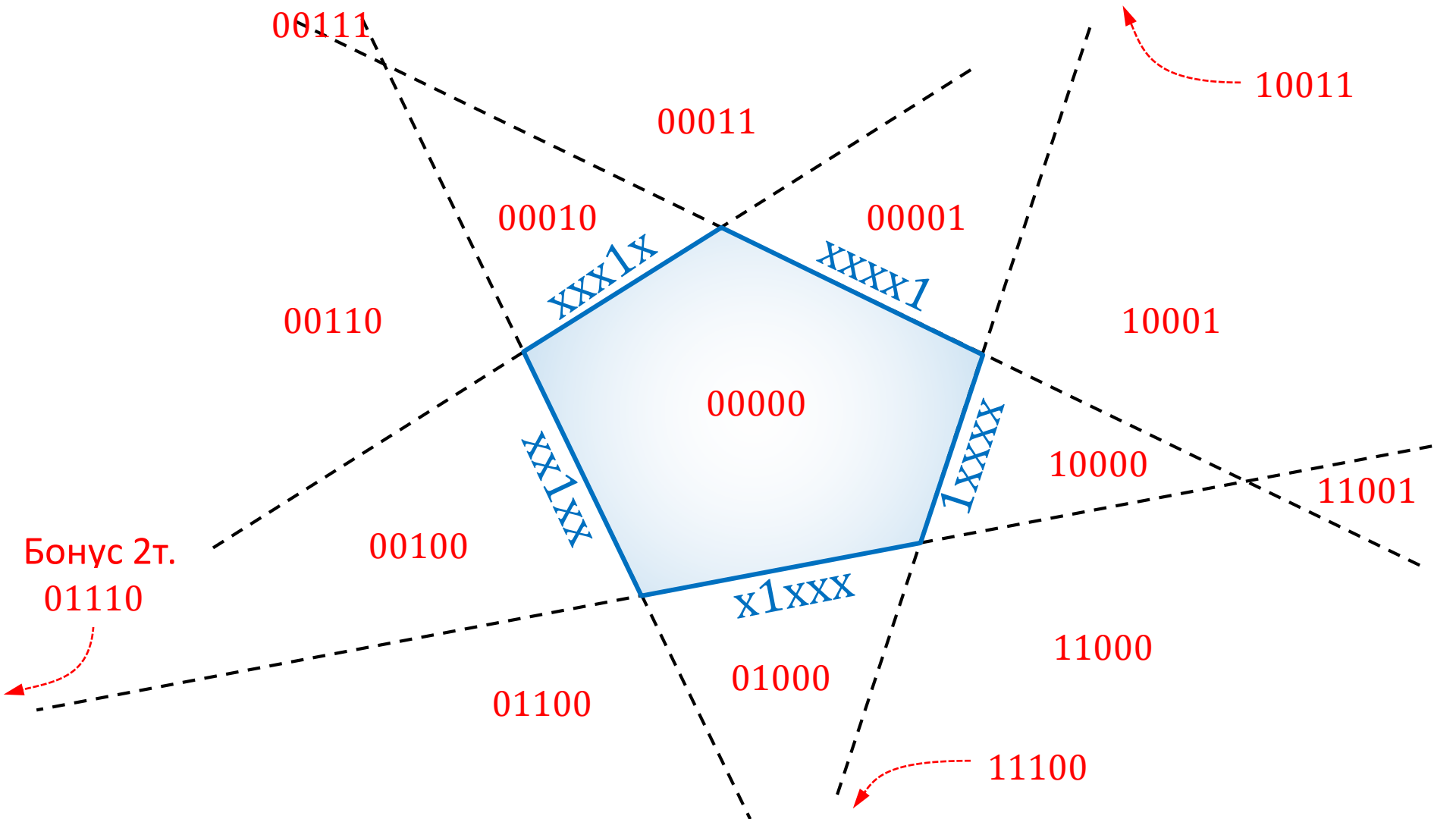


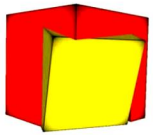


Многоъгълно изрязване

Приложение на алгоритъма на Коен-Съдърленд за многоъгълна зона

- Работи се с толкова битови числа, колкото ъгълен е многоъгълникът
- Отново централната зоната е с нулев код $000 \ldots 0$
- Броят зони зависи не само от броя стени, но и от ориентацията им


00111
00011
10011
00010
00001
xxx1x
xxxx1
00110
10001
00000
xx1xx
1xxxx
10000
11001
Бонус 2т.
00100
01110
x1xxx
11000
01000
01100
11100



Свързаност на зоните

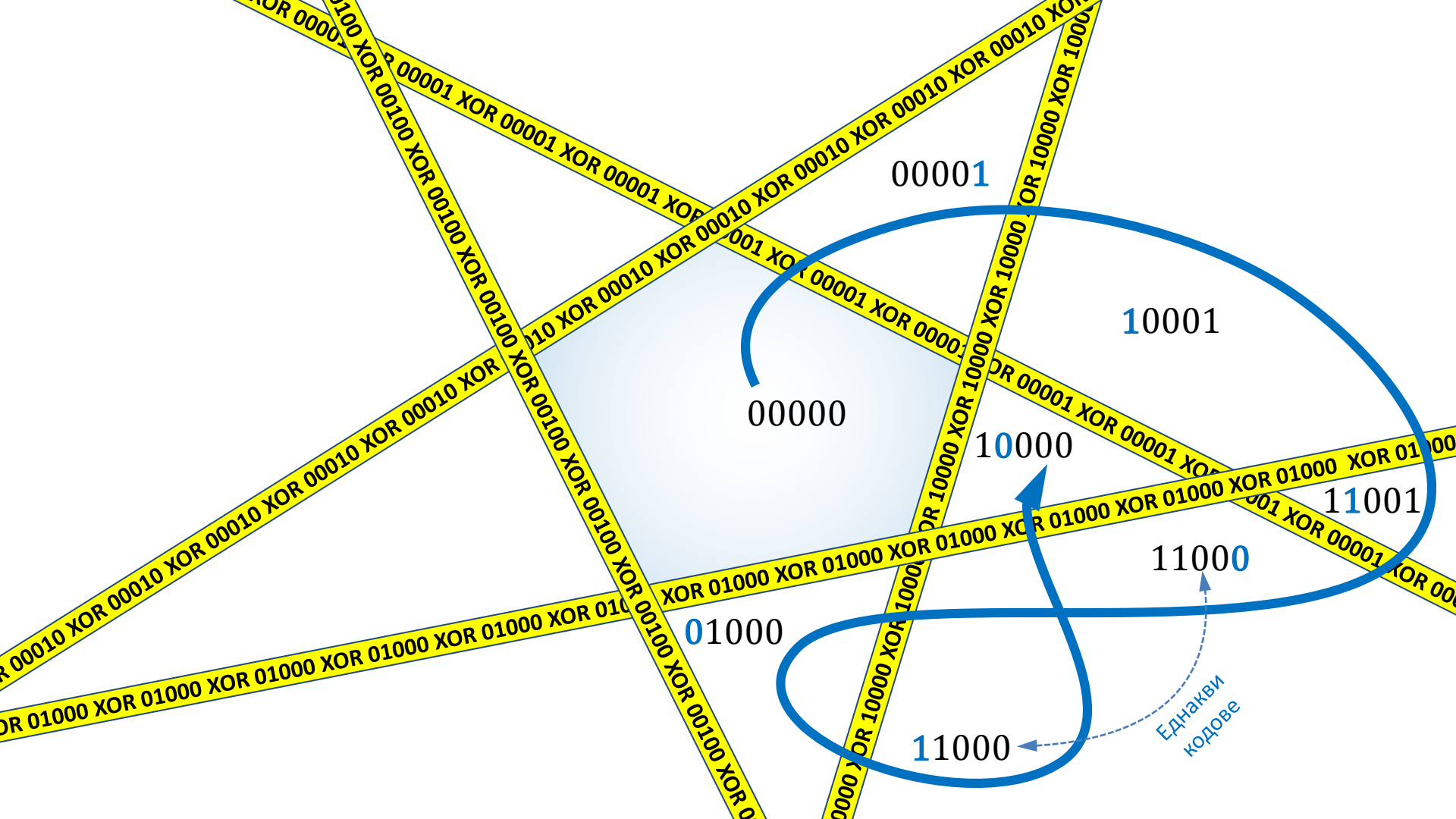
Кодове на две съседни зони

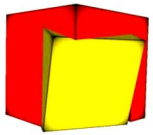
- Различават се по един бит
- Този бит е на правата между тях

Обхождане на зони

- Тръгваме от централната зона с код 0
- При всяко пресичане на права, xorваме съответния бит, т.е. $0 \to 1$ и $1 \to 0$

  (Ако сме умни, ще минем без $1 \to 0$. Защо? Още 1 точка бонус)


00001
10001
00000
10000
11001
11000
01000
11000
Еднакви кодове



В 3D

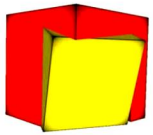
Алгоритъм на Коен-Съдърленд за многостени

- Изрязване на отсечка от многостен
- Всеки бит определя равнина, а не права

Пример

- За изрязване спрямо куб ще трябват 6-битови маски

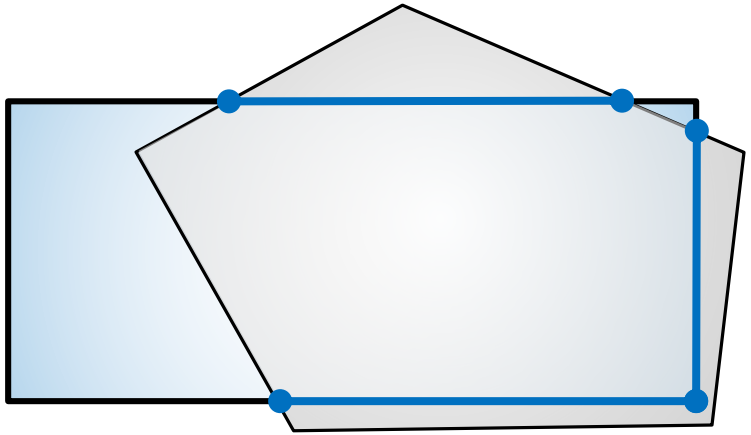
Отсичане на многоъгълник

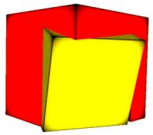


Отсичане на многоъгълник

Основна разлика

- Отсичането може да доведе до включването на отсечки, които не са по контура на многоъгълника



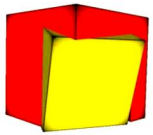


Алгоритми

Много алгоритми за отсичане на многоъгълник

- На Съдърленд-Ходжман (Sutherland-Hodgman)
- На Вайлер (Weiler)
- На Лиан(г)-Барски (Liang-Barsky)
- На Мейлот (Maillot)
- На Вати (Vatti)
- На Грайнер-Хорман (Greiner-Hormann)

Алгоритъм чрез отсичане на отсечка по Коен-Съдърленд



Първичен алгоритъм

Вариант на алгоритъм

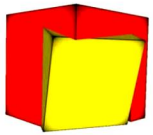
- Чрез отсичане на отсечка по Коен-Съдърленд

Особености

- Лесен за реализация
- Може да доведе до дублирани върхове
- Може да доведе до контурни точки
- Може да доведе до фалшиви страни

Основна идея на алгоритъма

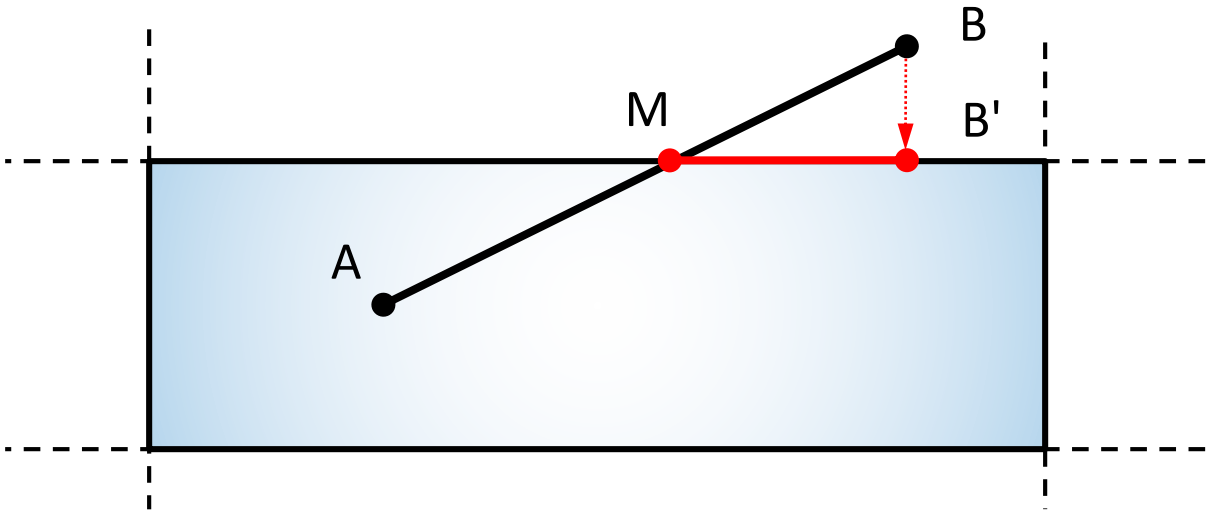
- Страните на многоъгълника се обхождат последователно
- Всяка се отсича чрез алгоритъма на Коен-Съдърленд
- Отсечените части не се изтриват, а се долепят до съответната отсичаща права



Пример

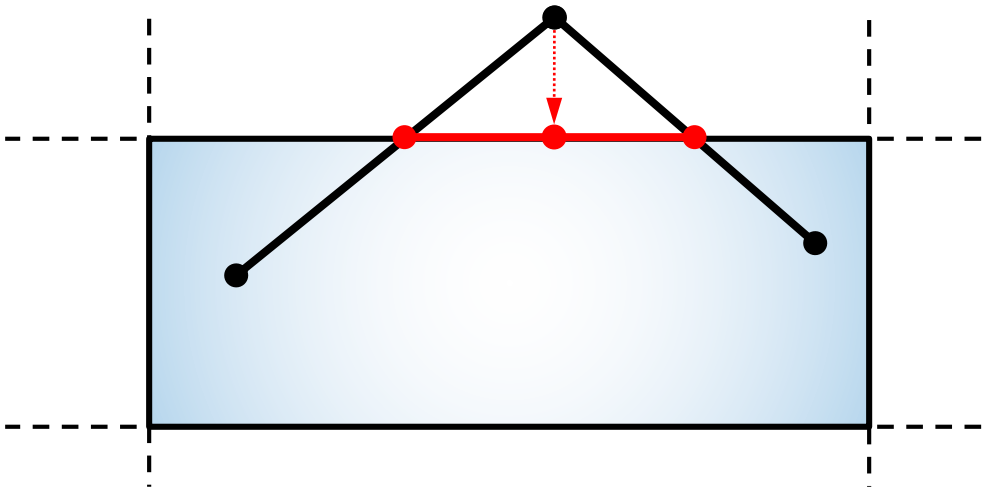
Обработване на отсечките

- AM се запазва, а MB се свлича до MB'



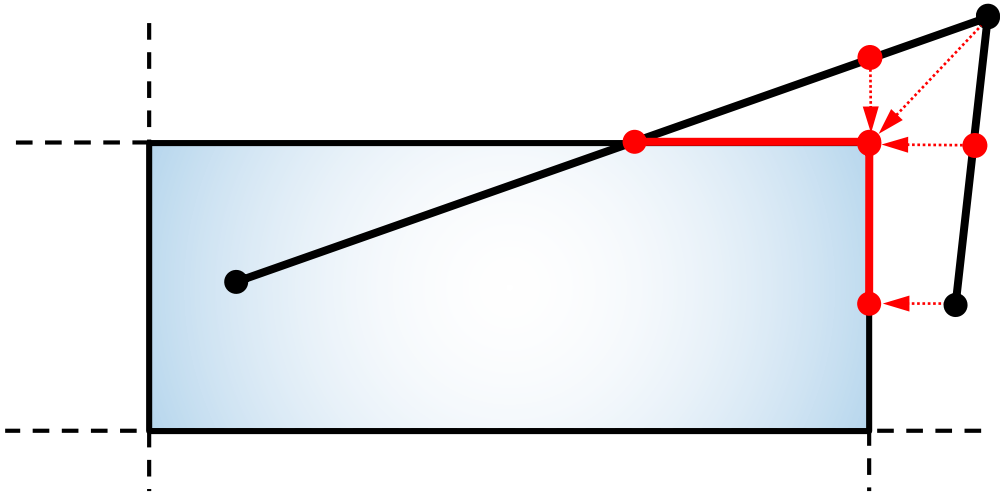
Получаване на контурни точки

- Могат да се елиминират в допълнителен пас
  (ако е необходимо)



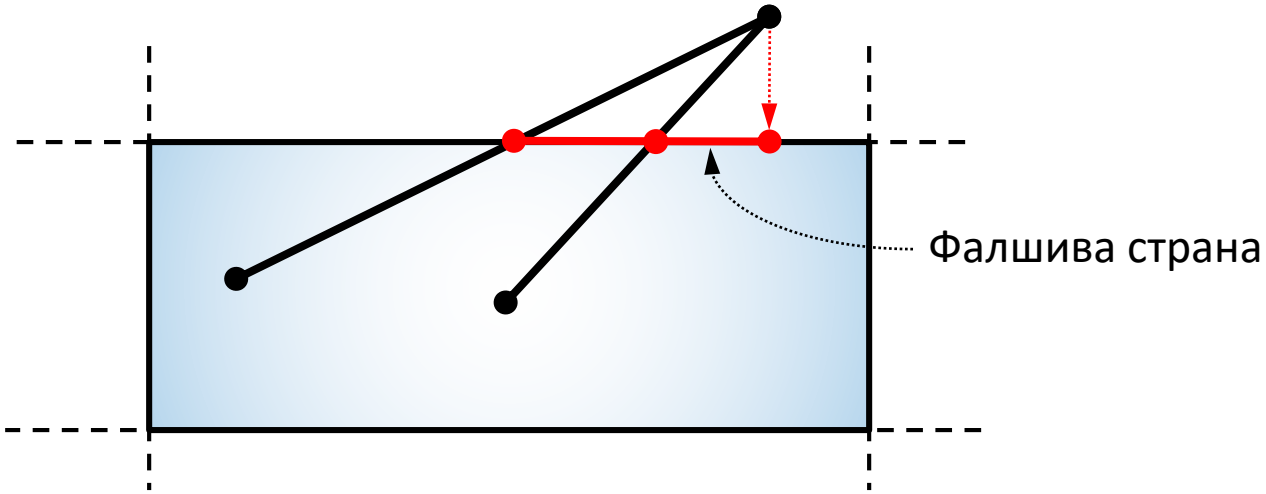
Получаване на дублирани върхове

- Могат да се елиминират в допълнителен пас
  (ако е необходимо)



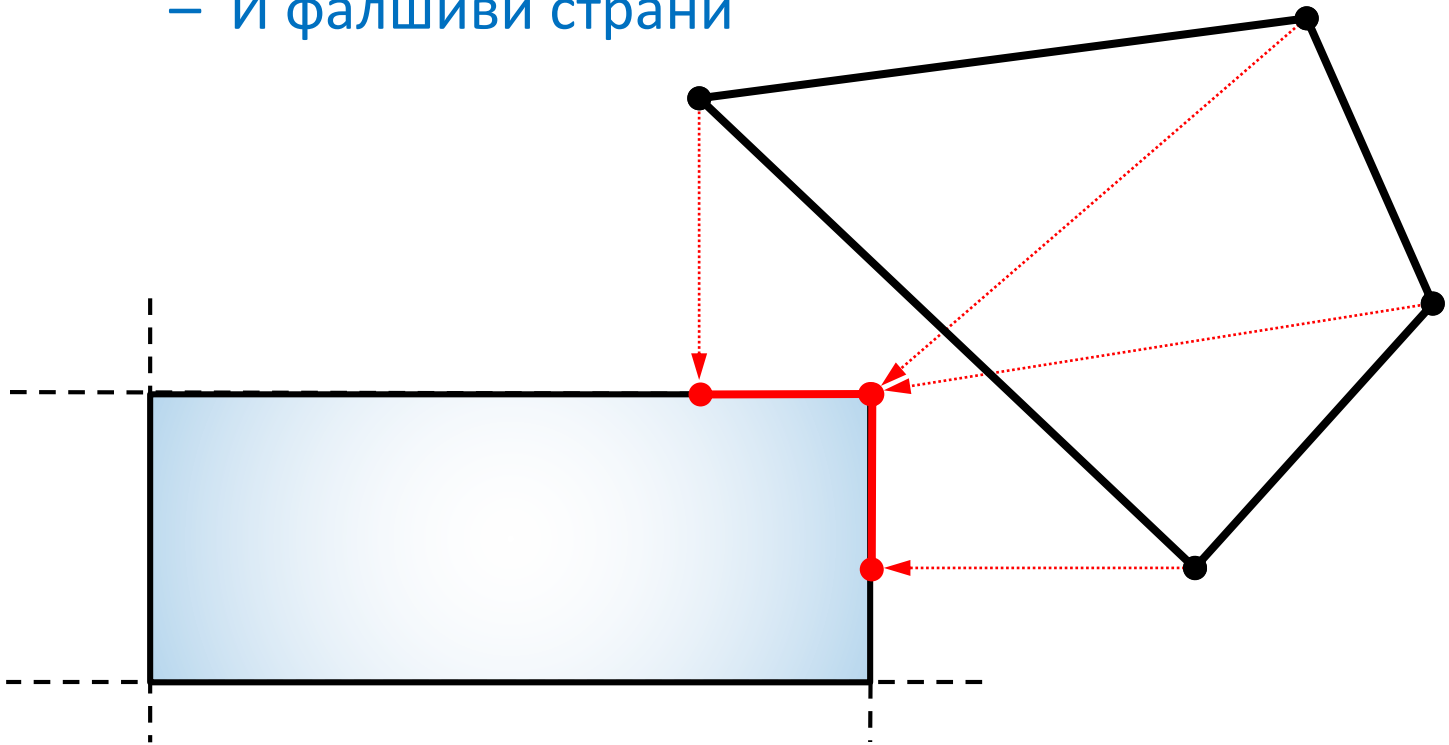
Получаване на фалшиви страни

- Могат да се елиминират в допълнителен пас
  (ако е необходимо)

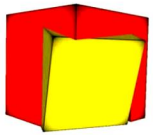


Гъдел при външни многоъгълници

- Само контурни и дублирани точки
- И фалшиви страни



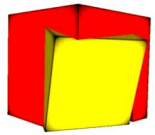
Алгоритъм на Съдърленд-Ходжман



Основна идея

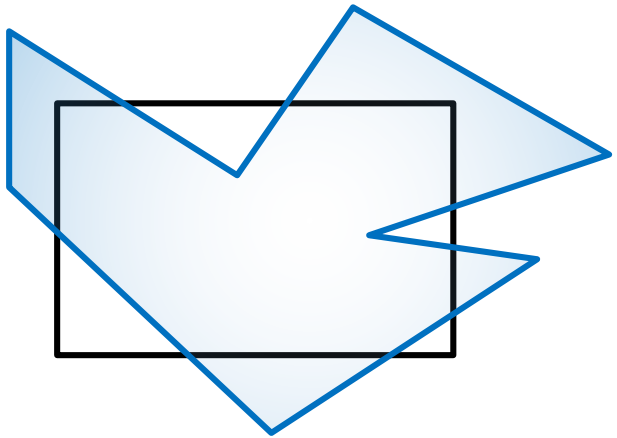
Основна идея на алгоритъма

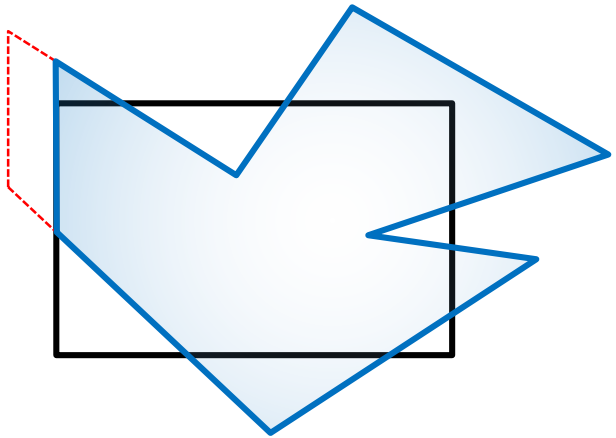
- Отсичаме многоъгълника по едната страна на видимата зона
- После по другата и т.н.
- За отсичането ни е нужно да различаваме вътрешност-външност на двойка точки



Отсичане по страни

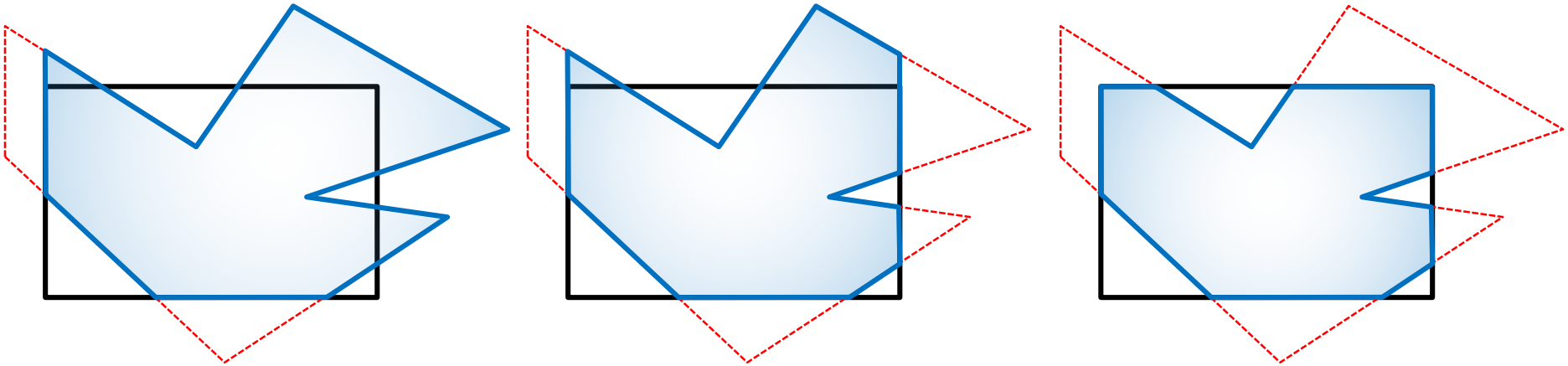
Отсичане отляво

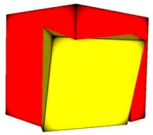




Отсичане по другите страни

- Долу, дясно, горе
- В произволен ред





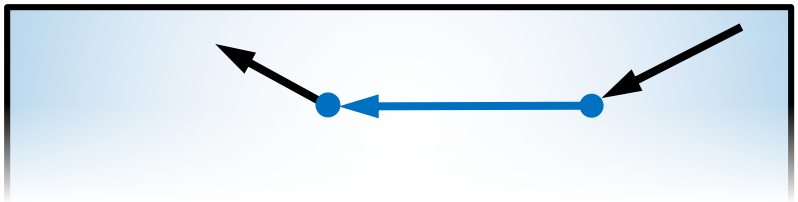
Как се отсича

Алгоритъм

- Обхождаме последователно страните на многоъгълника
- За всяка страна гледаме дали влиза или излиза от видимата зона
- По време на обхождането генерираме върховете на отсечения многоъгълник
- Четири прости правила

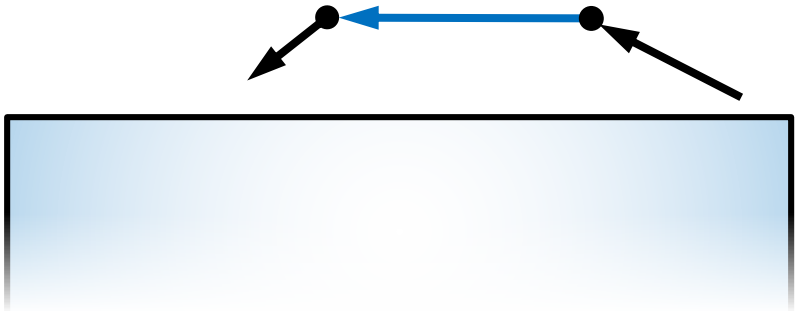
Двете точки са вътре

- И двете участват в резултата



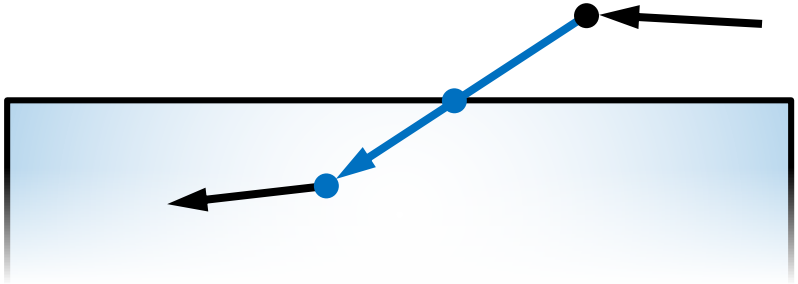
Двете точки са вън

- Нито една не участва в резултата



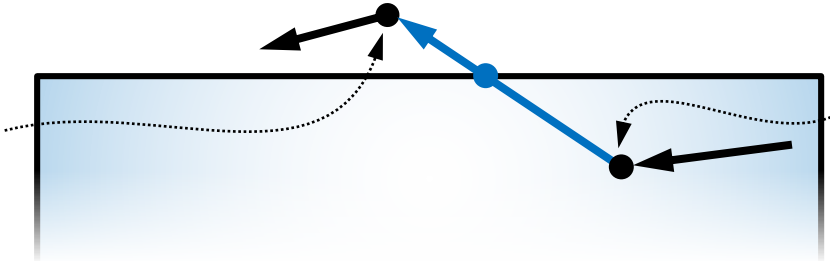
Влизаме във видимата зона

– Участват вътрешната и пресечната

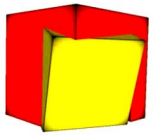


Излизаме от видимата зона

– Включваме само пресечната точка



Въпроси?



Повече информация

[**LUKI**]	стр. 86-109
[**AGO2**]	стр. 69-109
[**ALZH**]	гл. 4.4-4.5
[**KLAW**]	стр. 59-65
[**LASZ**]	стр. 122-128
[**MORT**]	стр. 296-299

А също и:

- Polygon Clipping
  http://www.codeguru.com/cpp/misc/misc/graphics/article.php/c8965/Polygon-Clipping.htm

Край