СОФИЙСКИ УНИВЕРСИТЕТ
„СВ. КЛИМЕНТ ОХРИДСКИ“

Растеризация

ТЕМА №8

Съдържание

Тема 8: Растеризация

- Растер
- Растеризация на отсечка
- Рекурсивен алгоритъм
- Алгоритъм със закръгляне
- Алгоритъм на Брезенхам

Растер

Пиксели

Пиксел (*pixel, pic's element, picture's element*)

- Най-малък елемент на растера
- В някои страни му викат пел (*pel, picture element*)
- В текстурите е тексел (*texel*, *texture pixel*)
- В сензорите е сенсел (*sensel, sensor pixel*)
- В 3D растерите е воксел (*voxel, volumetric pixel*)

Какво е растеризацията?

Растеризация

- Изобразяване на векторен или параметричен обект
  в дискретна мрежа от пиксели

Особености

- Растеризацията е апроксимация
- Искат се бързи целочислени алгоритми
- Понякога се работи с подпиксели

Топология

- Растерното пространство е дискретно
- Част от геометричните правила вече не важат
  (или поне не са монополни)

Основни проблеми

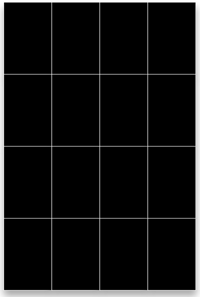
- Запазване на пропорции
- Запазване на разстояния
- Запазване на гладкост

Проблеми с пропорциите

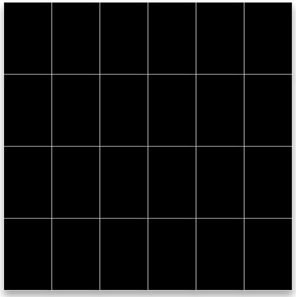
- Пикселът има размери
- Понякога пикселът не е „квадратен“

Пример с квадрат

- Съотношение на размерите на пиксел (аспект) 1:1.5



Неправилен квадрат
4x4 пиксела



Правилен квадрат 6x4 пиксела

Проблем с разстоянията

– Разстоянието не е уникално

Пример с правоъгълен триъгълник

– Всеки от катетите е 9 пиксела

– Хипотенузата също е 9 пиксела
(Евклиде, Питагоре, извинявайте!)

Три разстояния

Три дефиниции на разстояния

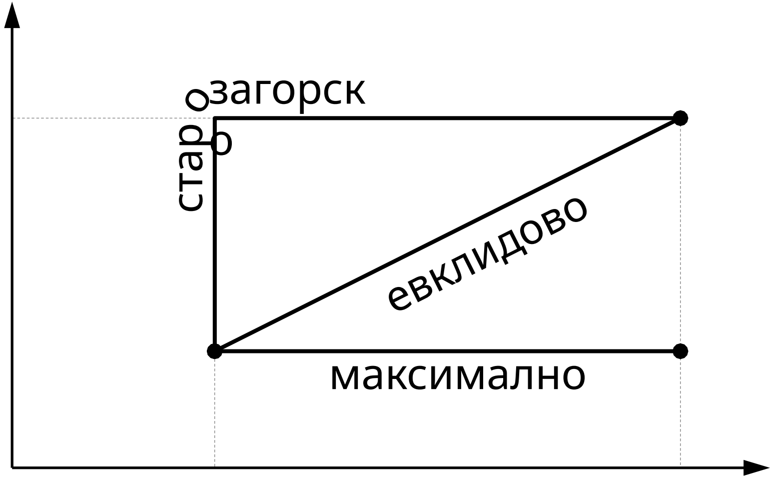
1. Евклидово: $r = \sqrt{\Delta x^2 + \Delta y^2}$
2. Таксиметрово или Манхатънско: $r_{sz} = |\Delta x| + |\Delta y|$
   (а защо не и Старозагорско)
3. Максимално: $r_{max} = \max(|\Delta x|, |\Delta y|)$

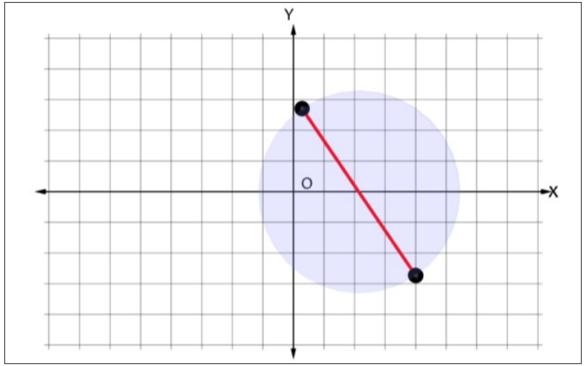
Употреба

1. И трите се ползват в КГ

Илюстрация

– Максималното разстояние е най-малко





Растеризиране на отсечка

Основна задача

Основната задача на растеризирането

- Да се намерят координатите на пикселите
- Най-доброто визуално приближение до отсечката

Алгоритми

- Рекурсивен алгоритъм с деление на 2
- Алгоритъм със закръгляне
- Алгоритъм на Брезенхам

Рекурсивен алгоритъм с целочислено деление на две

Рекурсивен алгоритъм

Начални данни

– Целочислени координати на два пиксела P и Q

Алгоритъм

Бонус 1т. Защо?

– Намира се пиксел $M = \frac{1}{2}(P+Q)$, но не $\frac{1}{2}P + \frac{1}{2}Q$

И още 1т.
Пак защо?

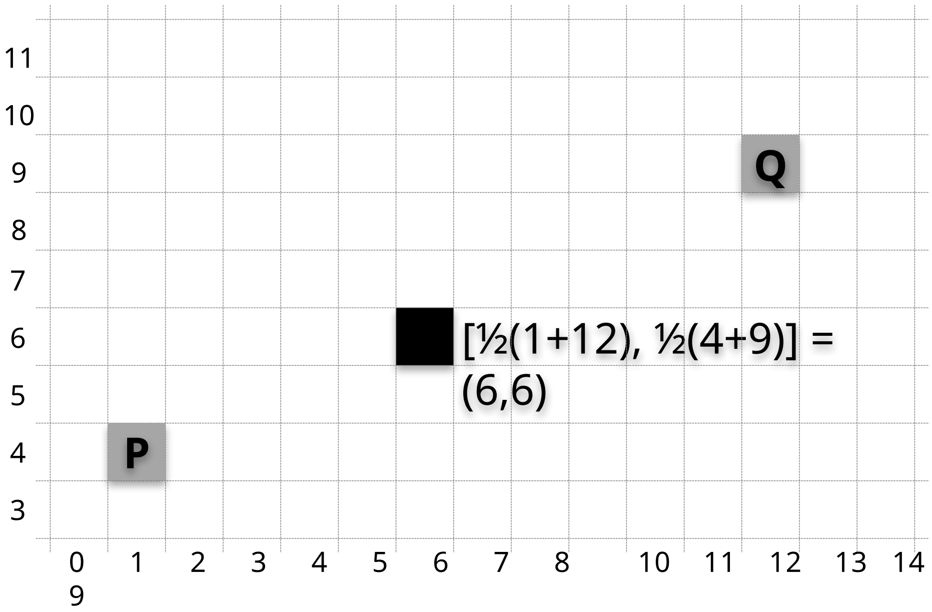
– Ако $\mathrm{M} \neq P$, повтаря се с PM

– Ако $\mathrm{M} \neq Q$, повтаря се с MQ

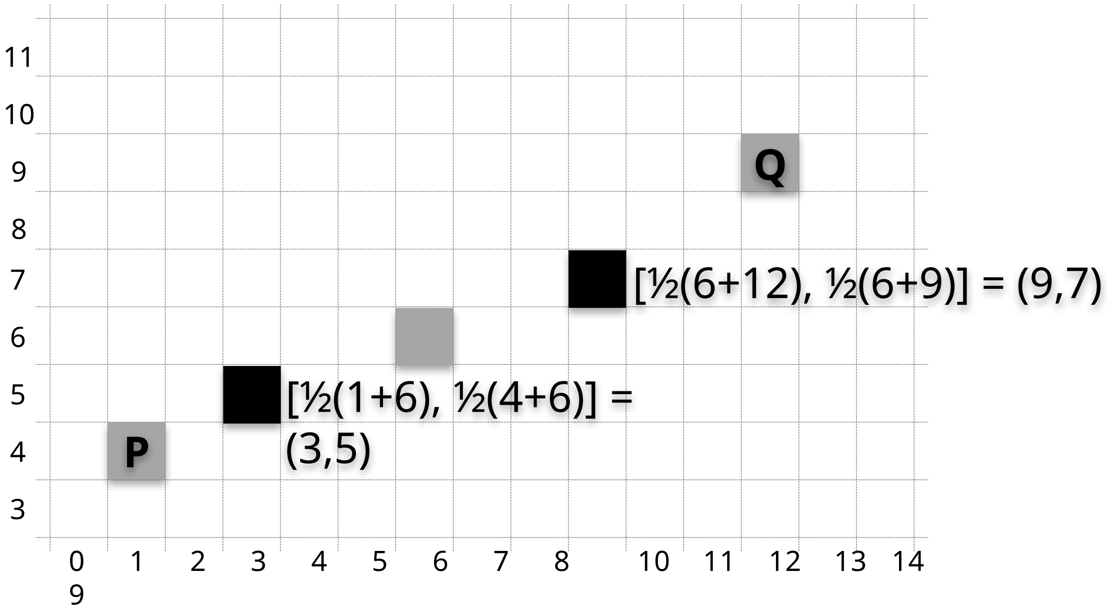
– Работим само с отсечки с $r_{max} > 1$

Пример P(1,4) и Q(12,9)

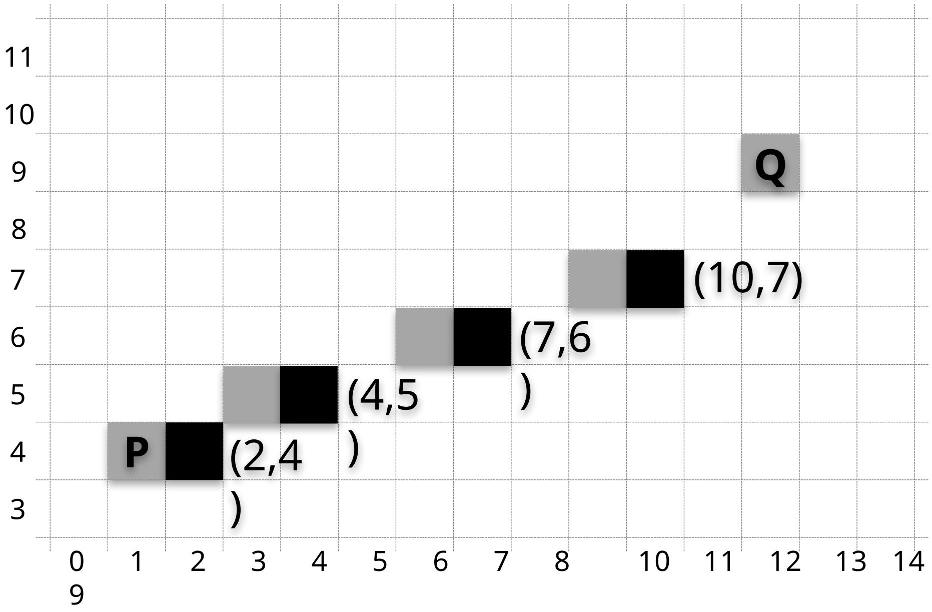
– Стъпка №1



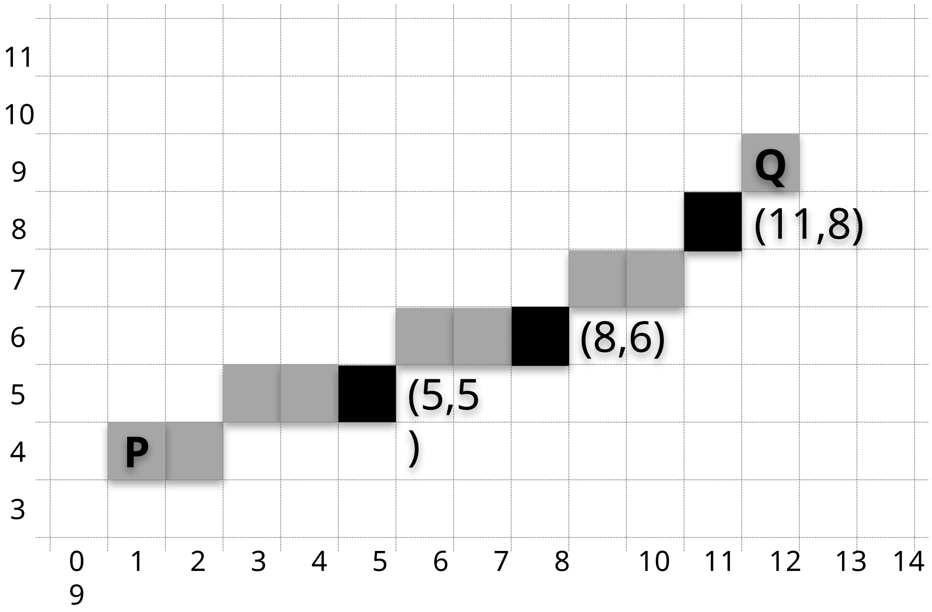
– Стъпка №2


11
10
9
8
7
6
5
4
3
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Q
P
[½(6+12), ½(6+9)] = (9,7)
[½(1+6), ½(4+6)] = (3,5)

– Стъпка №3

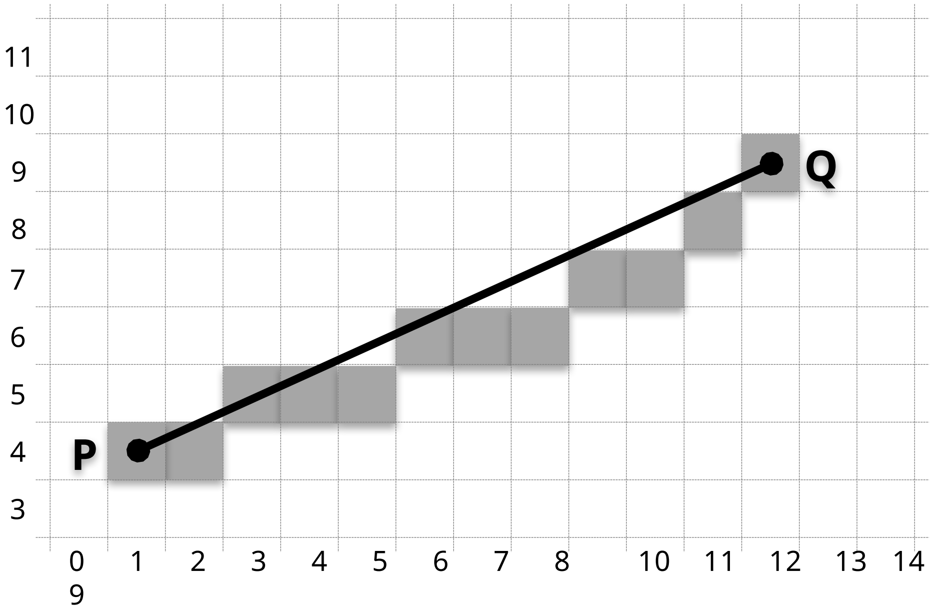


– Стъпка №4



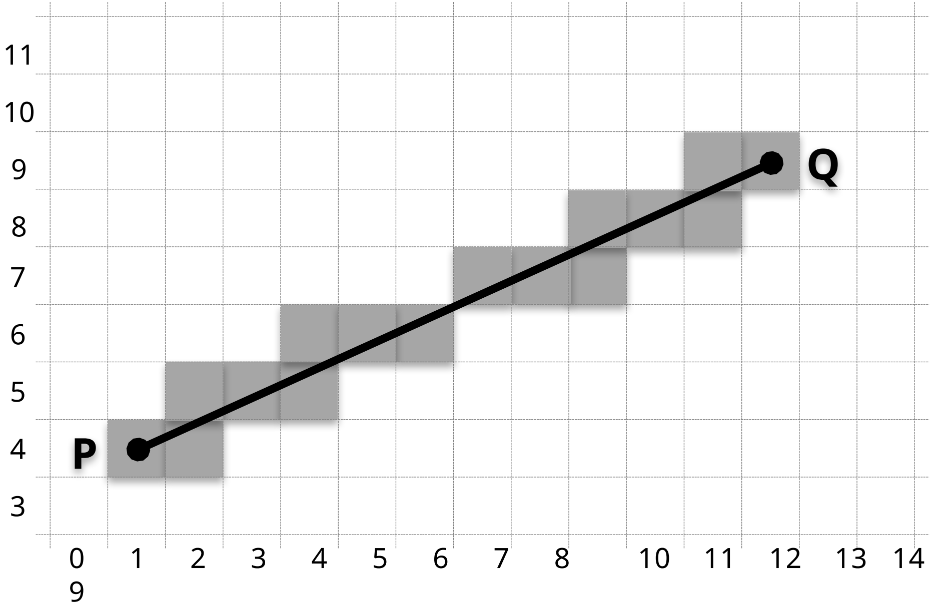
Проблем

– Външни пиксели, защото делим целочислено



А с реални числа?

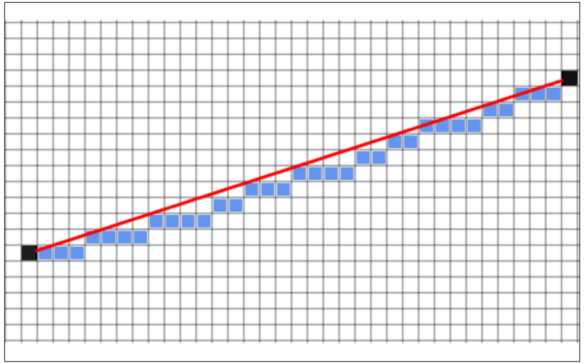
- Получава се по-добре
- Но не изглежда равномерно дебела

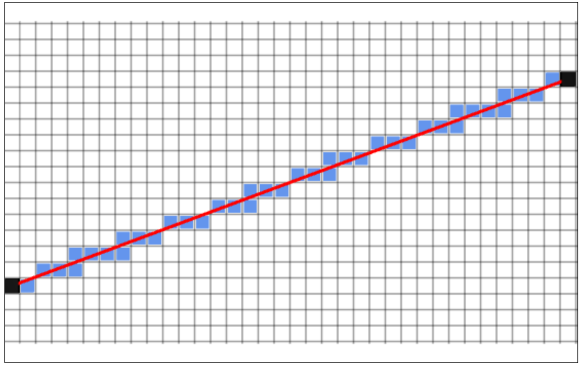


P = (1.00, 4.00)
Q = (12.00, 9.00)
M_1 = (6.50, 6.50)
M_2 = (3.75, 5.25)
M_3 = (9.25, 7.75)
M_4 ≈ (2.38, 4.63)
M_5 ≈ (1.69, 4.31)
M_6 ≈ (3.06, 4.94)
M_7 ≈ (5.13, 5.88)

Визуално сравнение

- Растеризация с целочислено деление
- Растеризация с реално деление





Алгоритъм
с единичен вектор

Алгоритъм с единичен вектор

Основни идеи

- С реални числа, за да не допусне външни пиксели
- Използва се наклонът на отсечката
- Получава пикселите последователно чрез единичния вектор

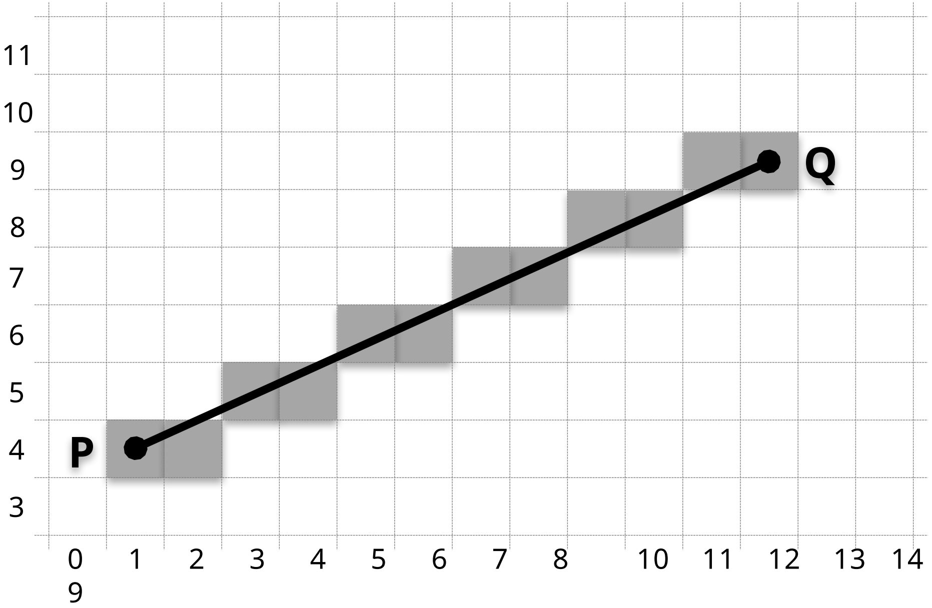
Определяне на единичен вектор

- Това е вектор $\vec{r}$ успореден на $\overrightarrow{PQ}$
- Има дължина $r_{max} = 1$ (не r и не r_{sz})
- Ако е нарисуван пиксел R_i, то следващият е $R_{i+1} = R_i + \vec{r}$
  (това гарантира, че R_{i+1} е съседен на R_i)
- Започва се от $R_0 = P$

Пример P(1,4) и Q(12,9)

- Намиране на $\vec{r}$

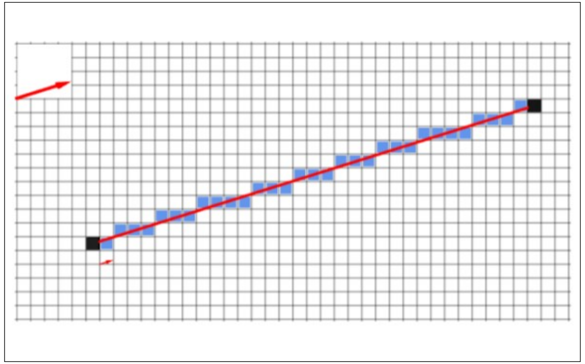
$$\begin{cases}\Delta x = 12 - 1 = 11 \\ \Delta y = 9 - 4 = 5\end{cases} \Rightarrow \ |\Delta x| \geq |\Delta y| \ \Rightarrow \ \vec{r} = \left(\frac{\Delta x}{|\Delta x|}, \frac{\Delta y}{|\Delta x|}\right) \approx (1{,}0.455)$$



R0 = (1, 4.000) = P
R1 ≈ (2, 4.455) ≈ (2, 4)
R2 ≈ (3, 4.910) ≈ (3, 5)
R3 ≈ (4, 5.365) ≈ (4, 5)
R4 ≈ (5, 5.820) ≈ (5, 6)
R5 ≈ (6, 6,275) ≈ (6, 6)
R6 ≈ (7, 6.730) ≈ (7, 7)
R7 ≈ (8, 7.185) ≈ (8, 7)
R8 ≈ (9, 7.640) ≈ (9, 8)
R9 ≈ (10, 8.095) ≈ (10, 8)
R10 ≈ (11, 8.550) ≈ (11, 9)
R11 ≈ (12, 9.005) ≈ Q

Алгоритъмът в действие

- Векторът е винаги с $|\Delta x|=1$ или $|\Delta y|=1$



Основни преимущества

- Правилно определя пикселите
- Броят стъпки е или $|\Delta x|$, или $|\Delta y|$

Основни недостатъци

- Ползва реални числа и това забавя
- Опасност от акумулирана грешка
- Различни стандарти за реални числа

Алгоритъм
на Брезенхам

Алгоритъм на Брезенхам

Обща информация

- Предложен от Брезенхам през 1965
- Използва само цели числа
- Използва събиране, изваждане и умножение по 2

За удобство

- Пикселите са възлите в мрежата
  (т.е. не са квадратчетата в мрежата)

Частен случай

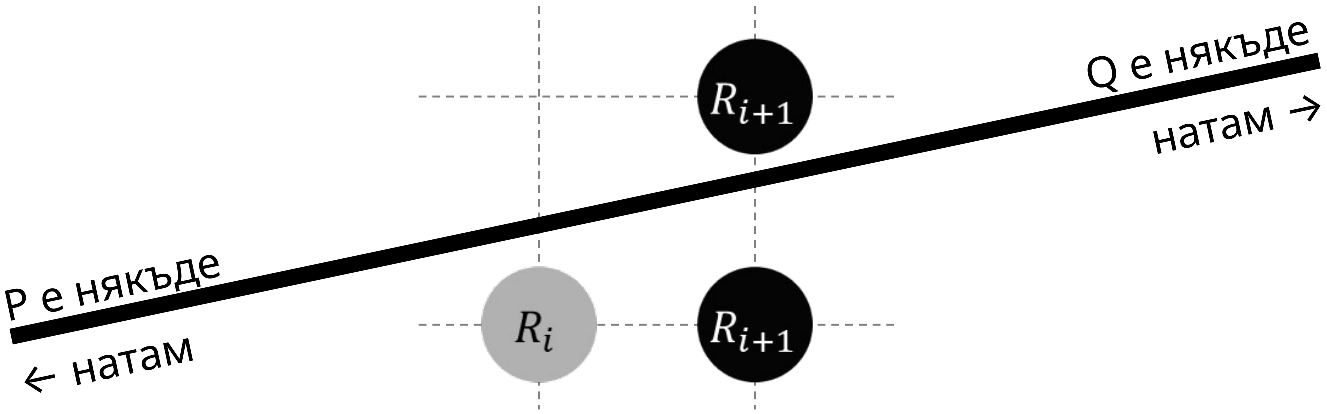
- Точка Q е вдясно от P и нагоре от нея, но повече вдясно, отколкото нагоре
- Т.е. $0 < \Delta y < \Delta x$ и ъгълът между $\overrightarrow{PQ}$ и $\overrightarrow{O_x}$ е [0°, 45°)

А другите случаи?

- Получават се от този чрез симетрия
  (сменят се знаци или се разменят X и Y)

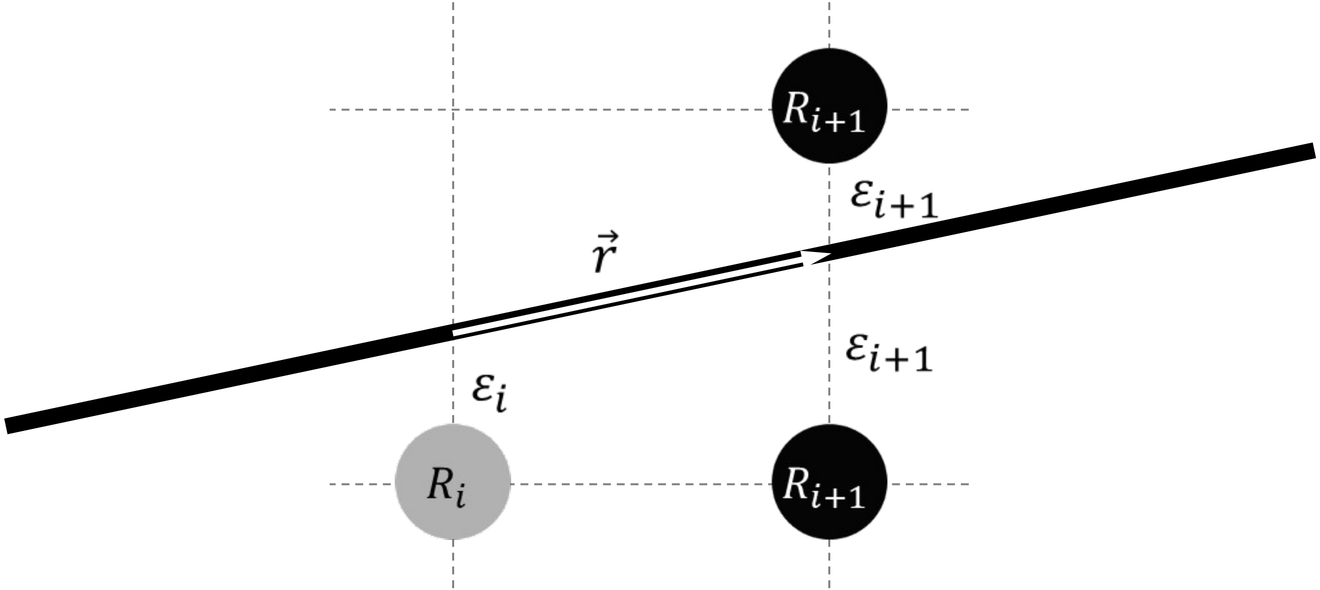
Междинна стъпка

- Вече е нарисувана точка R_i
- R_{i+1} ще бъде или вдясно от R_i, или диагонално нагоре-вдясно от R_i



Как се решава кое R_{i+1} се ползва?

- Отговор: което е по-близо правата
- Отклонение ε от реалния до нарисувания пиксел



Ако долното $\varepsilon_{i+1} \leq \frac{1}{2}$

- Следващата точка е долната R_{i+1}, а отклонението е $\varepsilon_{i+1} = \varepsilon_i + r_y = \varepsilon_i + \frac{\Delta y}{\Delta x}$
- Иначе следващата е горната R_{i+1}, а отклонението е $\varepsilon_{i+1} = \varepsilon_i + \frac{\Delta y}{\Delta x} - 1$ ⟵ Помислете защо, но не ми казвайте

И още

- Началното $\varepsilon_0 = 0$, но по принцип $\varepsilon_i \in \left[-\frac{1}{2}, \frac{1}{2}\right]$
- Да не се забравя, че $0 < \Delta y < \Delta x$, а векторът-стъпка е $\vec{r} = \left(1, \frac{\Delta y}{\Delta x}\right)$

Съблича се текстът, обличат се формулите

- Започва се с условието за избор на горното R_{i+1}: $\varepsilon_i + \frac{\Delta y}{\Delta x} > \frac{1}{2}$, т.е. $2\Delta x \varepsilon_i + 2\Delta y - \Delta x > 0$

Опростяване:

- Полага се $d_i = 2\Delta x \varepsilon_i + 2\Delta y - \Delta x$
- И се получава прекрасното условие $d_i > 0$
- Със следните свойства:
  $d_0 = 2\Delta x \varepsilon_0 + 2\Delta y - \Delta x = 2\Delta y - \Delta x$
  $\varepsilon_{i+1} - \varepsilon_i = \dfrac{d_{i+1} - d_i}{2\Delta x}$

Да не се забравя отклонението

- Разглеждат се и двата случая
  (за справка вижте слайда с оченце в горния десен ъгъл)
- Долно R_{i+1} при $d_i \leq 0$, горно R_{i+1} при $d_i > 0$
- Тогава $\varepsilon_{i+1} = \varepsilon_i + \frac{\Delta y}{\Delta x} - 1$, т.е. $\varepsilon_{i+1} - \varepsilon_i = \frac{\Delta y}{\Delta x} - 1$
- Изразява се ε чрез d: $\frac{d_{i+1} - d_i}{2\Delta x} = \frac{\Delta y}{\Delta x} - 1$
- Така се получава крайното $d_{i+1} = d_i + 2\Delta y - 2\Delta x$
  (d_{i+1} се получава от предходното d_i и е цяло число)

Сега като алгоритъм

Начало

– Започва се от точка $R_0 = P$ и $d_0 = 2\Delta y - \Delta x$

Стъпка

– Нарисувана е точка R_i

– Ако $d_i \leq 0$, точката R_{i+1} е тази вдясно,
а новото $d_{i+1} = d_i + 2\Delta y$

– Иначе точката R_{i+1} е тази горе-вдясно,
а новото $d_{i+1} = d_i + 2\Delta y - 2\Delta x$

Бонус 3т

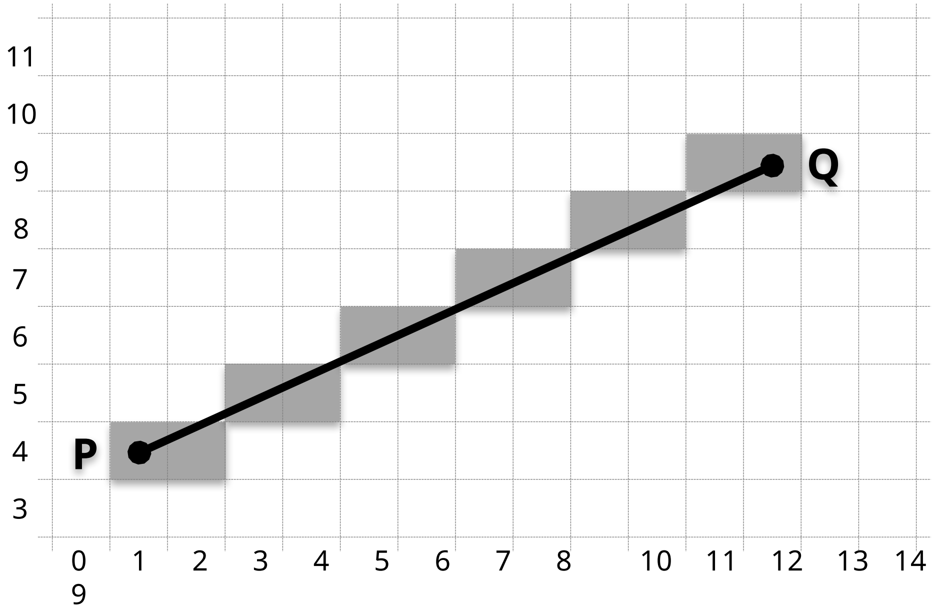
Защо не разделим на две?

Свойства на алгоритъма

- Началните данни са целочислени
  (координатите на точките P и Q)
- Операциите са само целочислени
  (събиране, изваждане и умножение по две)
- Всички получени резултати са целочислени
  (за всички операции има процесорни инструкции)

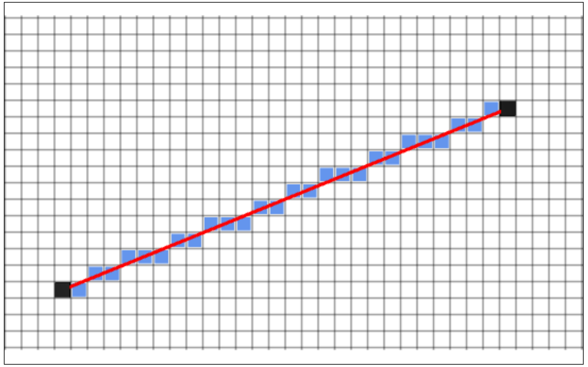
Пример P(1,4) и Q(12,9)

- Начални данни: $\Delta x = 11$ $\quad 2\Delta y = 10$ (при $d_i \leq 0$)
  $\Delta y = 5$ $\quad 2\Delta y - 2\Delta x = -12$ (при $d_i > 0$)
  $2\Delta y - \Delta x = -1$ (начално d_0)



R_0 = (1, 4) d_0 = -1
} +10
R_1 = (2, 4) d_1 = 9
} -12
R_2 = (3, 5) d_2 = -3
} +10
R_3 = (4, 5) d_3 = 7
} -12
R_4 = (5, 6) d_4 = -5
} +10
R_5 = (6, 6) d_5 = 5
} -12
R_6 = (7, 7) d_6 = -7
} +10
R_7 = (8, 7) d_7 = 3
} -12
R_8 = (9, 8) d_8 = -9
} +10
R_9 = (10, 8) d_9 = 1
} -12
R_{10} = (11, 9) d_{10}= -11
R_{11} = (12, 9) d_{11}= без значение

Алгоритъмът в действие



Въпроси?

Повече информация

LUKI стр. 27-37
AGO2 стр. 25-26
ALZH глава 4.2
KLAW стр. 43-55

А също и:

- Color Rasterizing primitives (стр. 2-14)
  http://alamos.math.arizona.edu/~rychlik/CourseDir/535/resources/LineDrawing.pdf
- Raster Algorithms (стр. 5-30)
  http://caig.cs.nctu.edu.tw/course/CG2007/slides/raster.pdf

Край