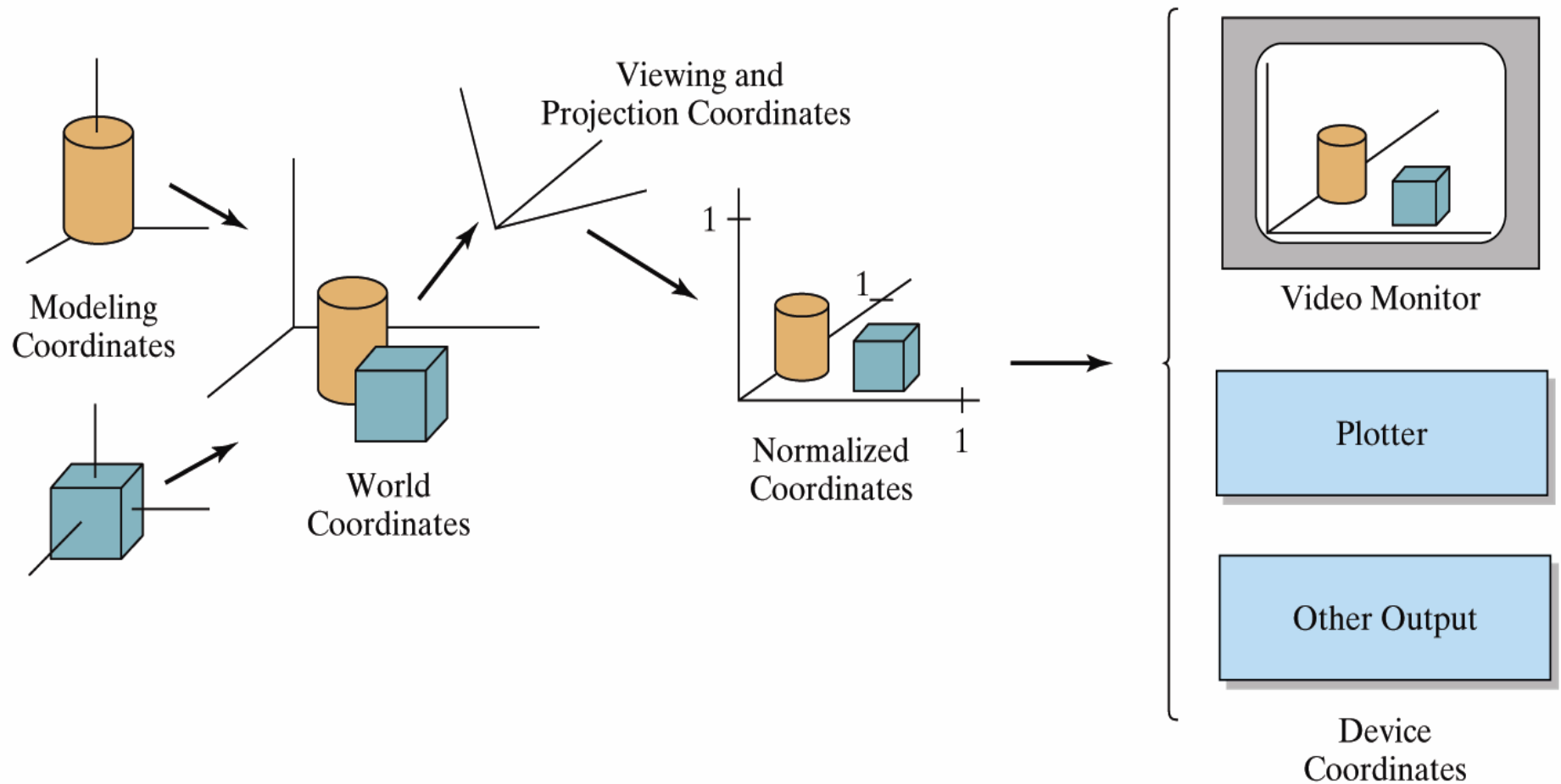

Компютърна графика

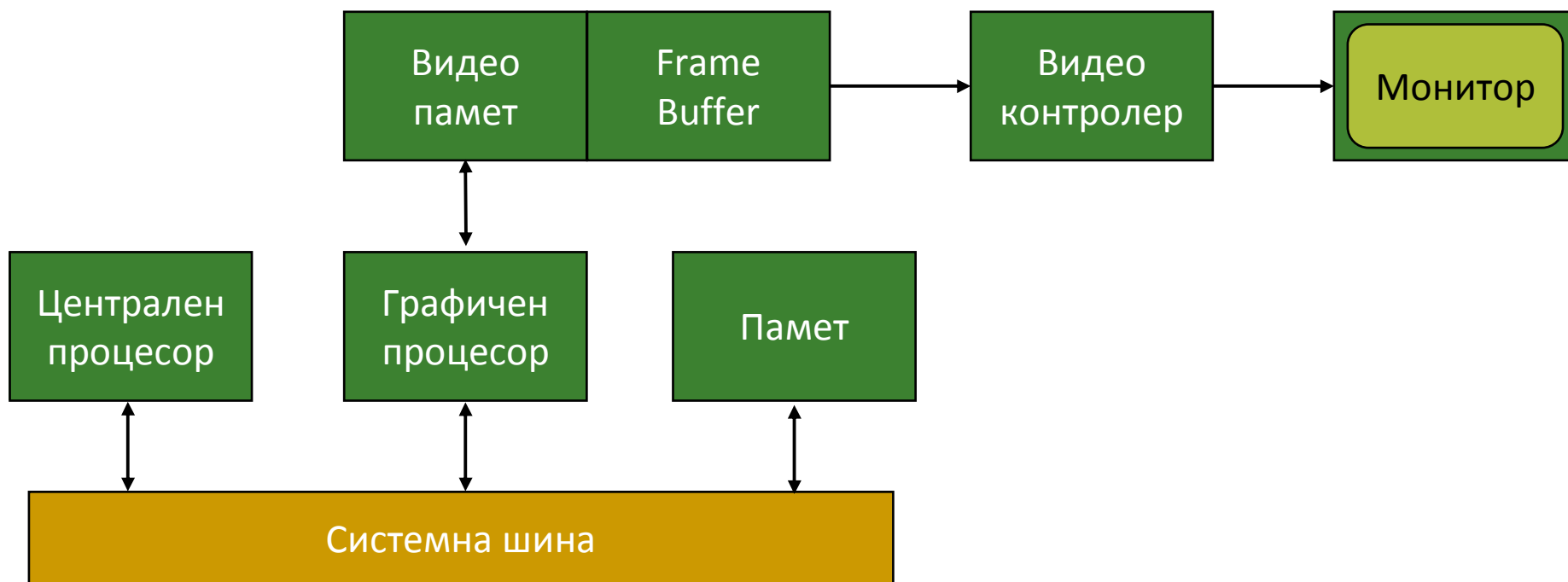
Основни алгоритми за растеризиране

доц. Милена Лазарова, кат. КС, ФКСУ

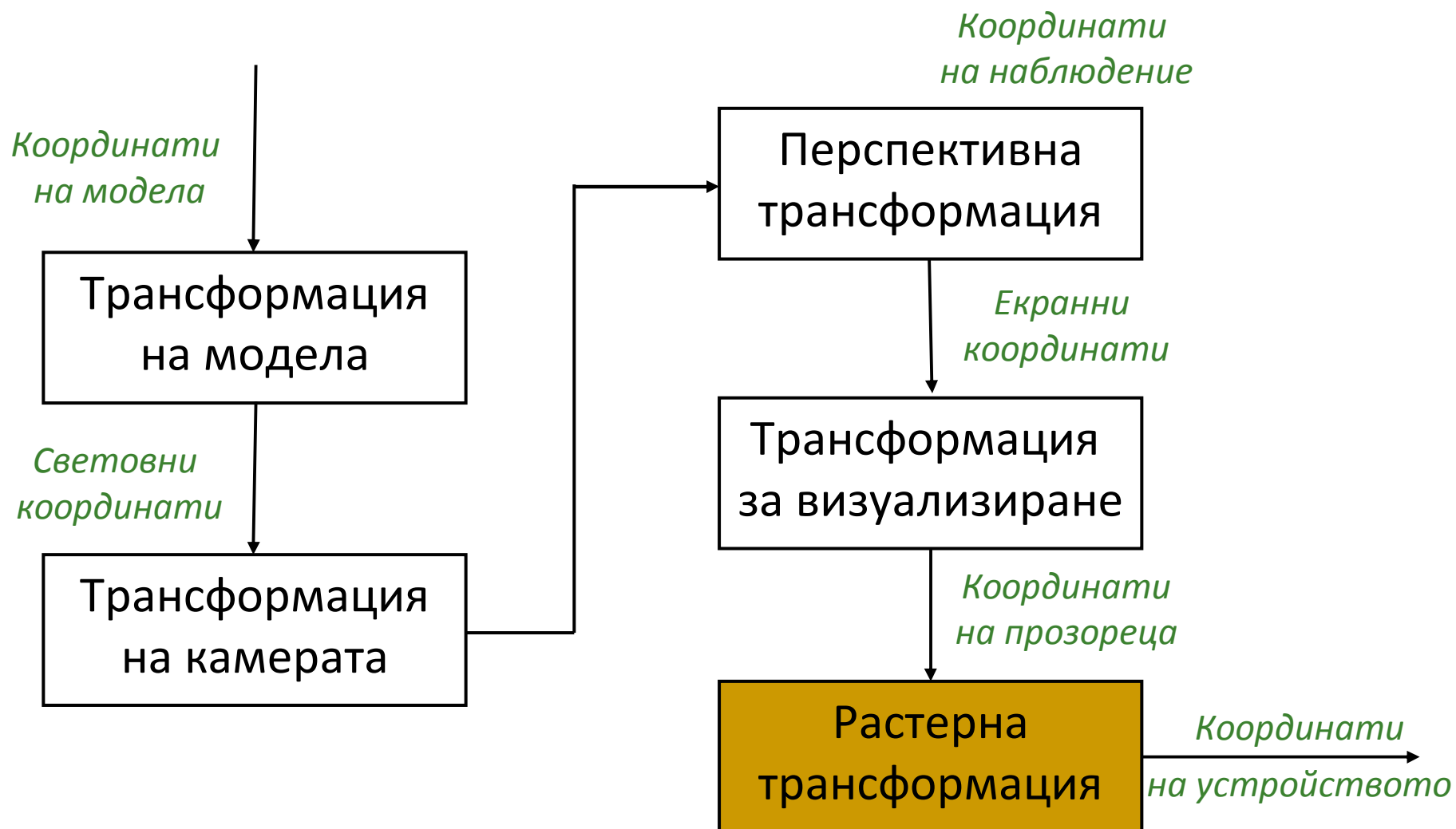
Компютърна графика



Архитектура на графична система



Основен графичен конвейер



Растрерна трансформация

- Последната стъпка от графичния конвейер
 - растрерна трансформация
 - растеризиране (*scan conversion*)
 - преобразуване на геометрични примитиви в масив от пиксели, съхранявани в буферна памет преди визуализиране
- Изпълнява се след изрязването (clipping)
- Всички графични библиотеки изпълняват растеризирането в края на графичния конвейер
 - преобразуват се примитиви в пиксели на екрана
 - отчитат се и други характеристики като осветеност

Растрерна трансформация

- *Да се изчертае линия между две точки в растрен екран*
- Некоректно поставена задача
 - не се задава какви могат да бъдат крайните точки
 - не се дефинира какво означава “да се изчертае”
 - не се определя какво е “линия” в растрен модел
 - не се формулира как ще се измерва качество на различните алгоритми за изчертаване

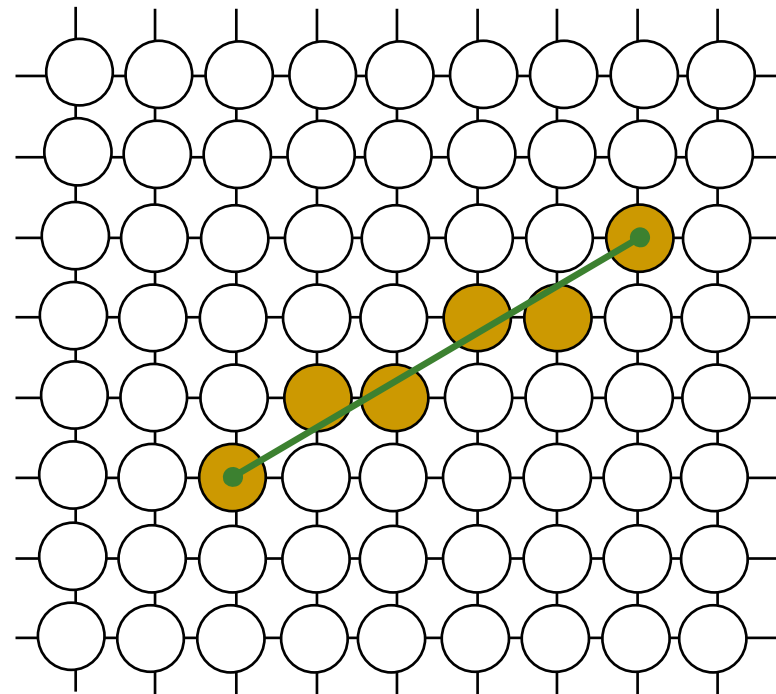
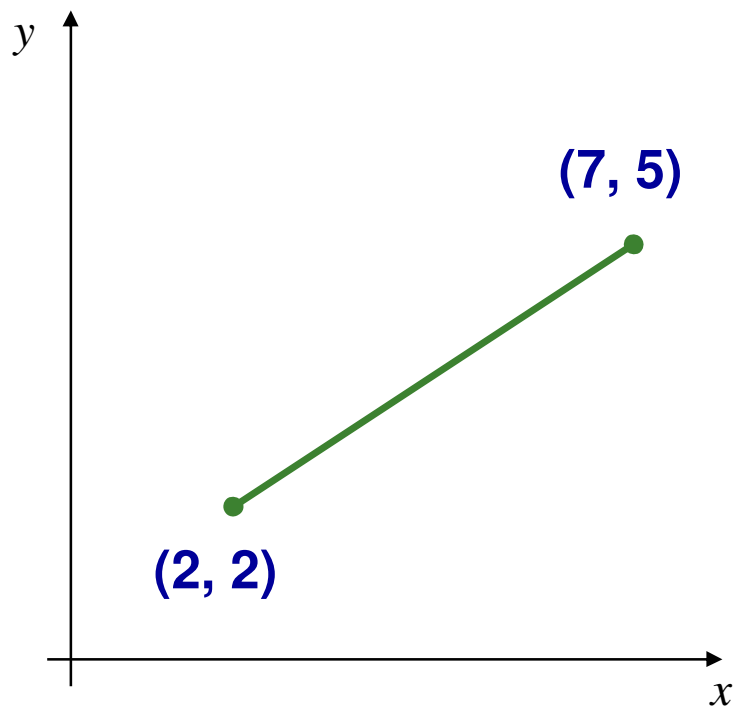
Изчертаване на линия

- **Формулиране на задачата**

- Дадени са две точки P и Q с целочислени координати в 2D координатна система
- Да се определи кои пиксели от растерен образ трябва да се визуализират за да се представи линеен сегмент с единична дебелина с начална точка P и крайна точка Q

Изчертаване на линия

- Как да се определи кои пиксели от растерния образ да се изчертаят?



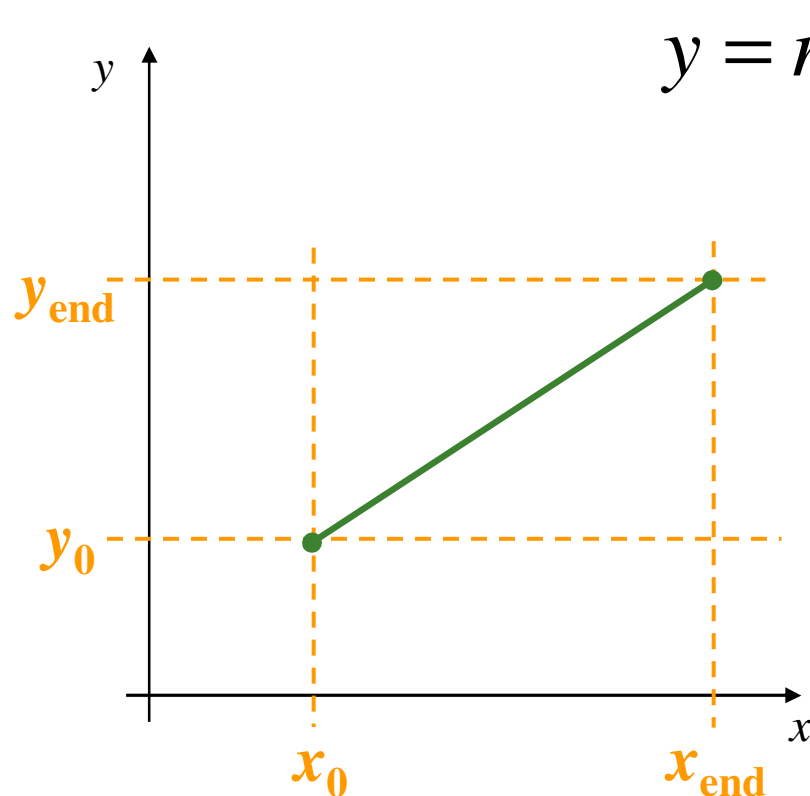
Изчертаване на линия

■ Изисквания

- линията да има колкото е възможно по-правилна форма (права)
 - да се избегнат “нащърбвания” (“jaggies”)
- да се изчертава бързо!
 - обикновено сцените съдържат голям брой линии

Изчертаване на линия

■ Декартово уравнение на права



$$y = m \cdot x + b$$

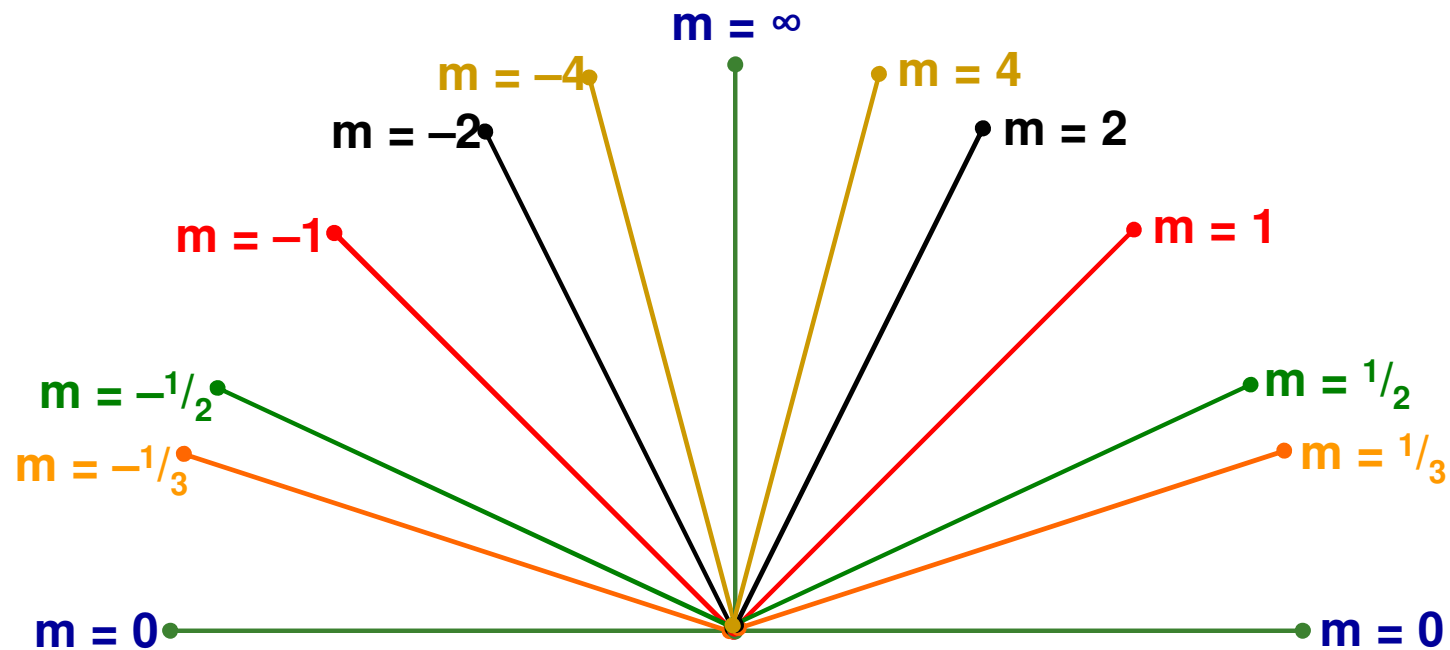
където ъгловият коефициент m и отрезът b се определят като

$$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$

$$b = y_0 - m \cdot x_0$$

Изчертаване на линия

- Ъгловият коефициент m определя наклона на правата



Изчертаване на линия

■ Алгоритъм на грубата сила (“brute force”)

два варианта

- по стойностите на x да се определят стойностите на y

$$y = m \cdot x + b$$

- по стойностите на y да се определят стойностите на x

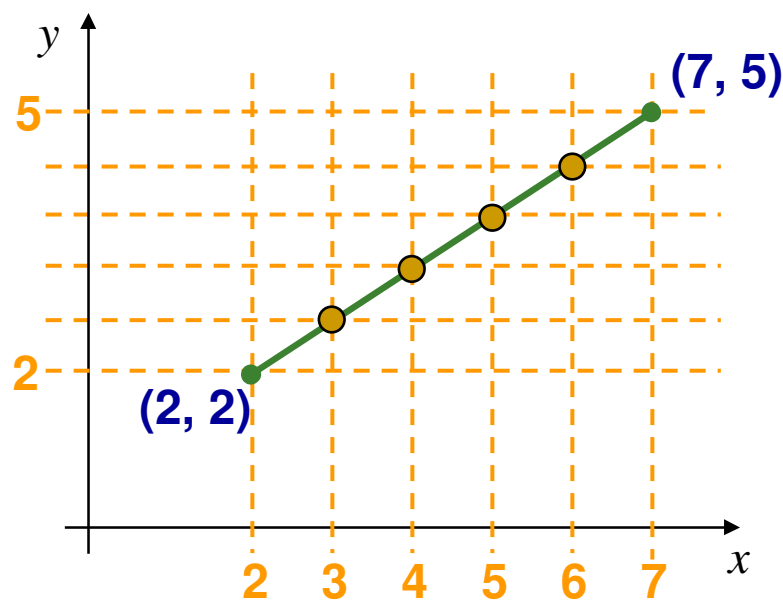
$$x = \frac{y - b}{m}$$

- където $m = \frac{y_{end} - y_0}{x_{end} - x_0}$ $b = y_0 - m \cdot x_0$

Алгоритъм на грубата сила

■ Пример

- изчисляват се m и b : $m = \frac{5-2}{7-2} = \frac{3}{5}$ $b = 2 - \frac{3}{5} \cdot 2 = \frac{4}{5}$
- за всяка стойност на x се определя стойността на y



$$y(3) = \frac{3}{5} \cdot 3 + \frac{4}{5} = 2\frac{3}{5}$$

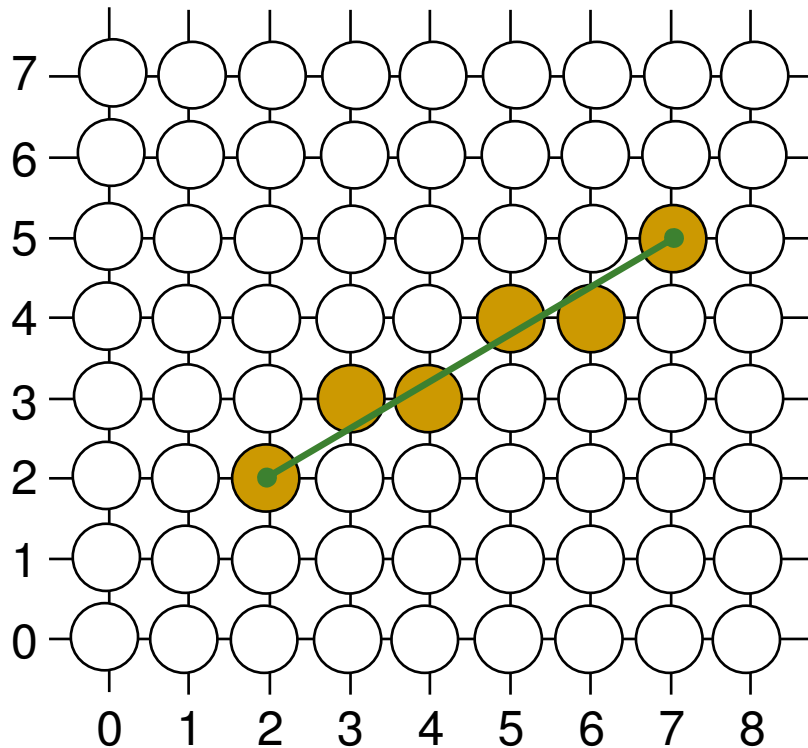
$$y(4) = \frac{3}{5} \cdot 4 + \frac{4}{5} = 3\frac{1}{5}$$

$$y(5) = \frac{3}{5} \cdot 5 + \frac{4}{5} = 3\frac{4}{5}$$

$$y(6) = \frac{3}{5} \cdot 6 + \frac{4}{5} = 4\frac{2}{5}$$

Алгоритъм на грубата сила

- Изчислените стойности се **закръгляват** и се изчертават



$$y(3) = 2\frac{3}{5} \approx 3$$

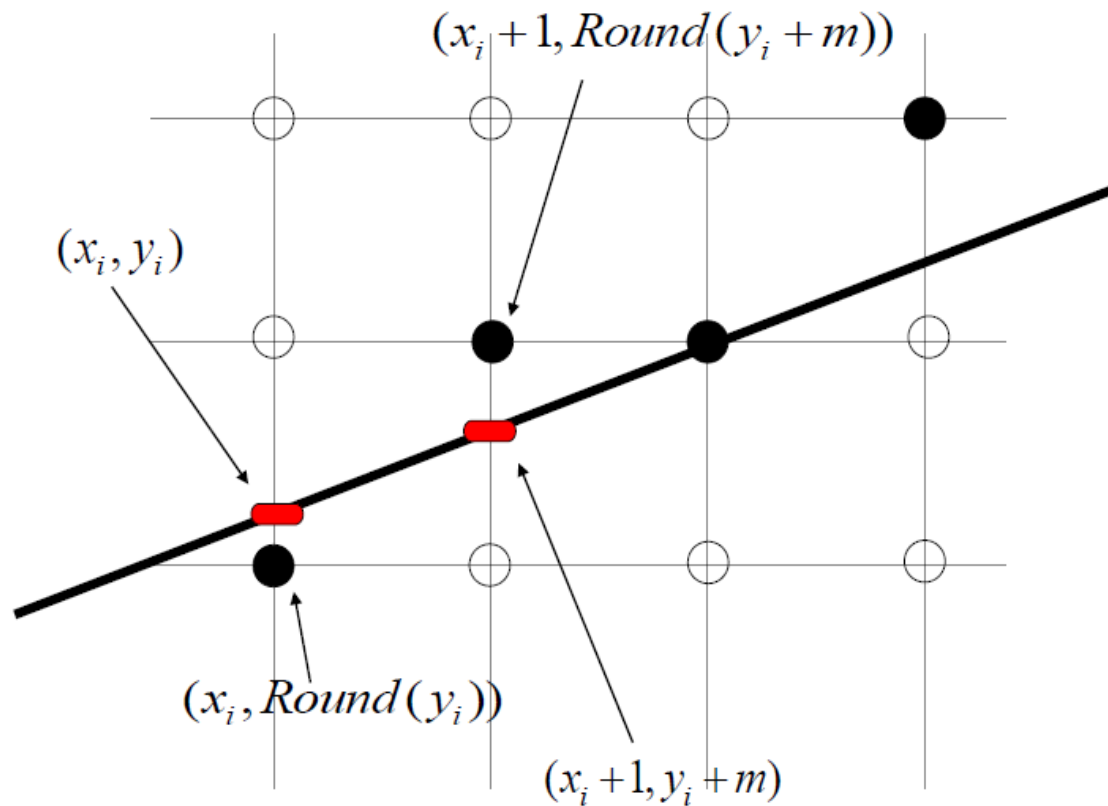
$$y(4) = 3\frac{1}{5} \approx 3$$

$$y(5) = 3\frac{4}{5} \approx 4$$

$$y(6) = 4\frac{2}{5} \approx 4$$

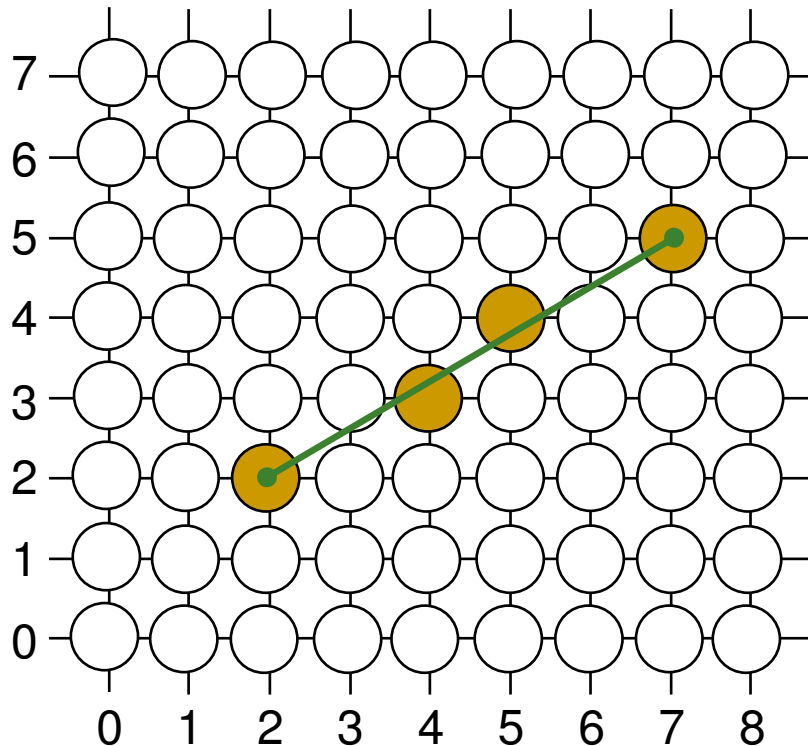
Алгоритъм на грубата сила

- Изчислените стойности се **закръгляват** и се изчертават



Алгоритъм на грубата сила

- Ако по стойностите на y се изчислят и закръглят стойностите на x , то се получава



$$x(3) = 3 \frac{2}{3} \approx 4$$

$$x(4) = 5 \frac{1}{3} \approx 5$$

- по-лош резултат
- как да се изчертае линията (спрямо x или спрямо y) се определя от наклона m

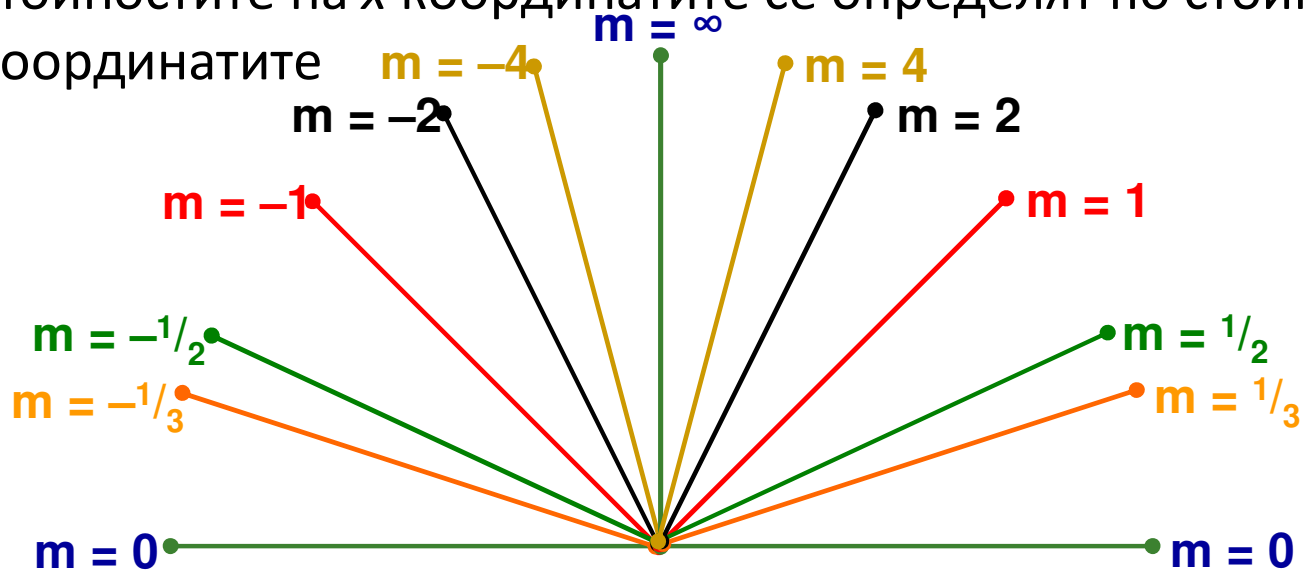
Алгоритъм на грубата сила

■ За права с наклон между -1 и 1

- стойностите на y координатите се определят по стойностите на x координатите

■ В противен случай

- стойностите на x координатите се определят по стойностите на y координатите



Алгоритъм на грубата сила

■ *Алгоритъм на грубата сила*

□ бавен алгоритъм

- определянето на стойностите на y изисква *умножение* на m и x
- определянето на пикселите за изчертаване изисква *закръгляване*

Изчертаване на линия

■ Алгоритъм *Digital Differential Analyzer (DDA)*

- за ускоряване на изчисленията се използва **инкрементален** подход
- стойността на координатата y_{k+1} се определя на базата на стойността на координатата y_k

- *Differential Analyzer* е физическа машина, създадена през 1930 в MIT от Vannevar Bush с цел решаване на диференциални уравнения

Алгоритъм DDA

- В примера за алгоритъма на грубата сила изчертаването на линията е на базата на точките

(2, 2)

(3, 23/5)

(4, 31/5)

(5, 34/5)

(6, 42/5)

(7, 5)

- *При увеличаване на стойността на x координатите с 1, стойностите на y координатите се увеличават със стойността на наклона на правата*

Алгоритъм DDA

- За права линия с наклон между -1 и 1
 - започвайки от началната точка на линията стойността на x координатата се инкрементира с 1 , а стойността на съответната y координата се определя като

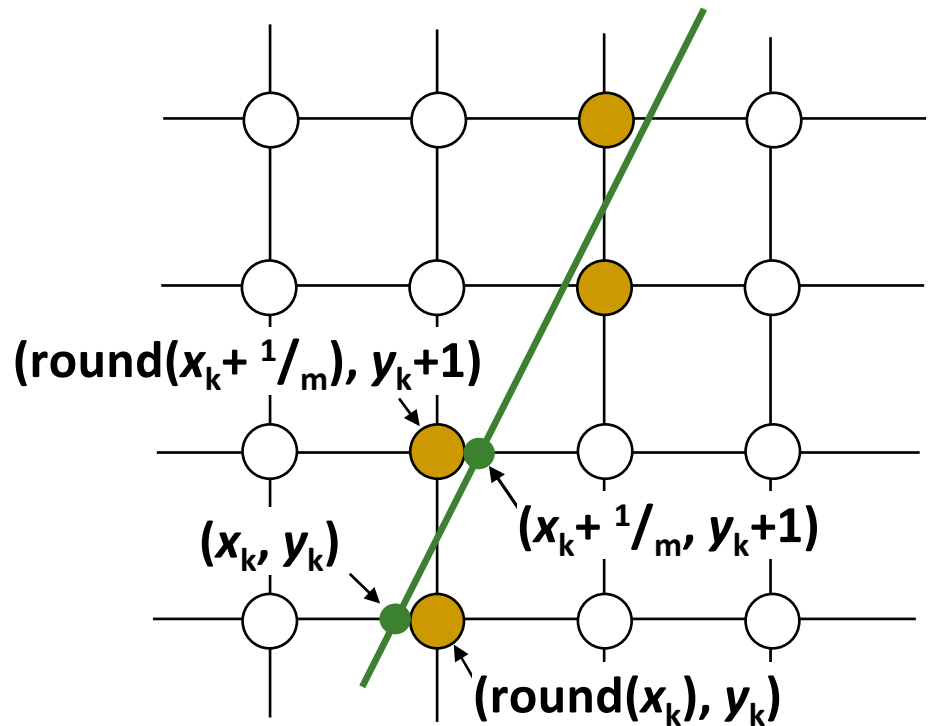
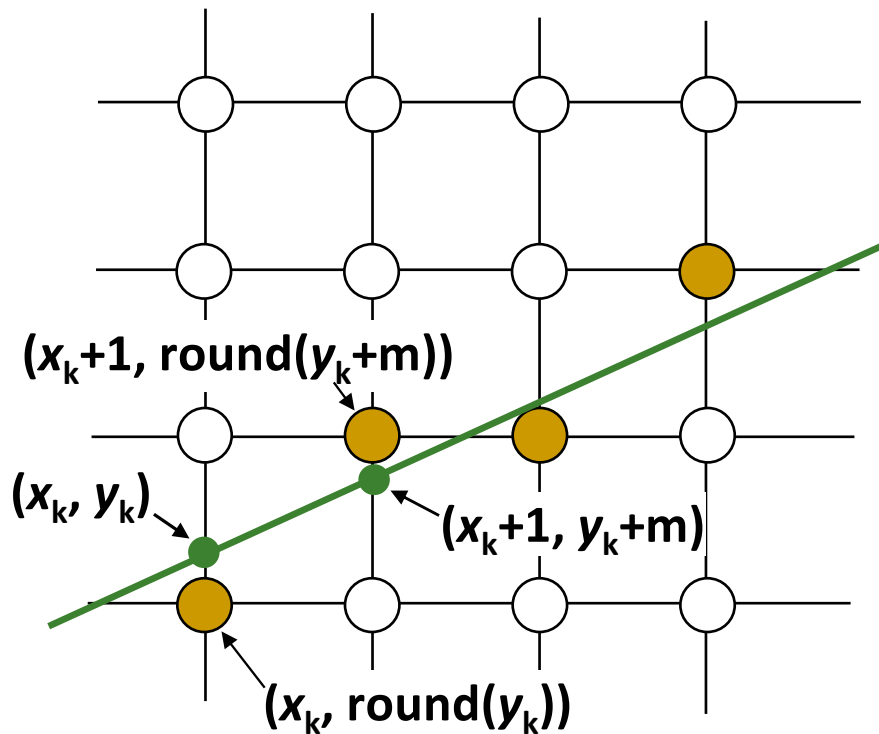
$$y_{k+1} = y_k + m$$

- За права с наклон по-голям от 1 или по-малък от -1
 - y координатата се инкрементира с 1 и се изчислява съответната x координата като

$$x_{k+1} = x_k + \frac{1}{m}$$

Алгоритъм DDA

- Изчислените с алгоритъма DDA стойности се **закръгляват** за да се определят координатите на пикселите за изчертаване



Алгоритъм DDA

■ *Предимства*

- алгоритъмът DDA е много по-бърз от алгоритъма на грубата сила
 - не изисква умножение

■ *Недостатъци*

- натрупването на грешки от закръгляване може да доведе до отклоняване на растеризираната линия от векторното ѝ представяне
- закръгляването и операциите с плаваща запетая са изчислително сложни

Алгоритъм DDA

```
// DDA Line Algorithm  
// Assume x0 < x1
```

```
void Line(int x0, int y0,  
         int x1, int y1) {  
    int x, y;  
    float dy = y1 - y0;  
    float dx = x1 - x0;  
    float m = dy / dx;  
  
    y = y0;  
    for (x = x0; x < x1; x++) {  
        WritePixel(x, Round(y));  
        y = y + m;  
    }  
}
```

закръгляването е изчислително сложно
и води до натрупване на грешка

тъй като наклонът се определя като отношение,
вертикалните линии са специфичен случай

Алгоритъм DDA

- Определяне на следващ пиксел

- специални случаи*

- *хоризонтална линия*

- изчертава се пиксел P и се увеличава x координатата му с 1 за да се определи следващия пиксел

- *вертикална линия*

- изчертава се пиксел P и се увеличава y координатата му с 1 за да се определи следващия пиксел

- *диагонална линия*

- изчертава се пиксел P и се увеличават x и y координатите му с 1 за да се определи следващ пиксел

Алгоритъм DDA

- Определяне на следващ пиксел
в общия случай
 - увеличава се x координатата с 1 и се определя точка, която е най-близка до линията
- *Как се измерва “близост”?*

Изчертаване на линия

- **Алгоритъмът на Брезенам**
 - Bresenham
 - Midpoint Line Algorithm
- Основно предимство
 - използва само **целочислена аритметика**

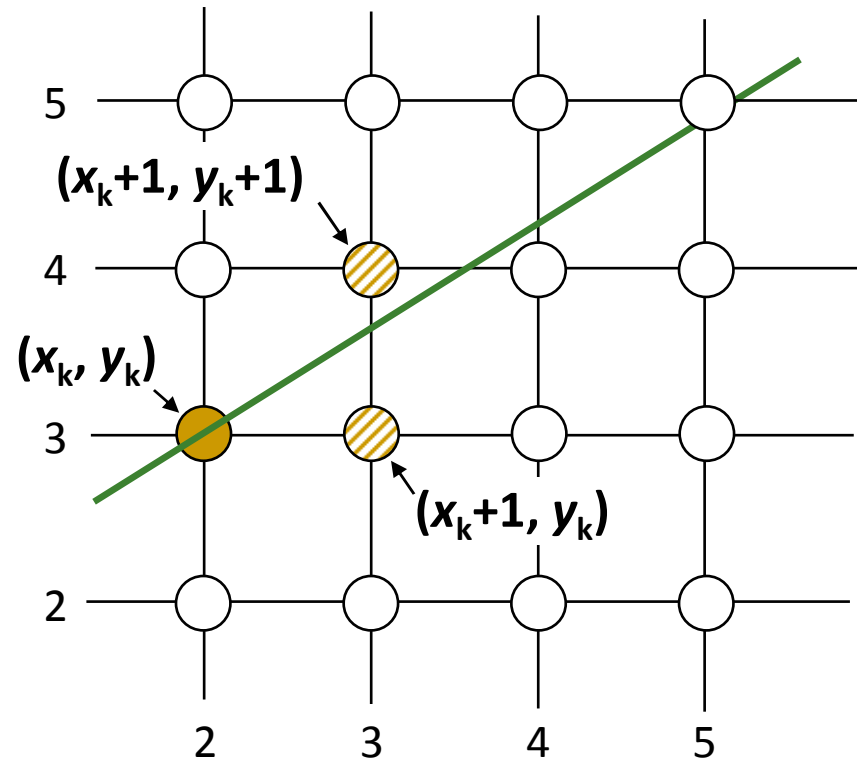
Алгоритъмът на Брезенам

■ Основна идея

- x координатата се увеличава с 1 и се избира една от две възможни y координати

- например от точка $(2, 3)$ трябва да се избере между $(3, 3)$ и $(3, 4)$

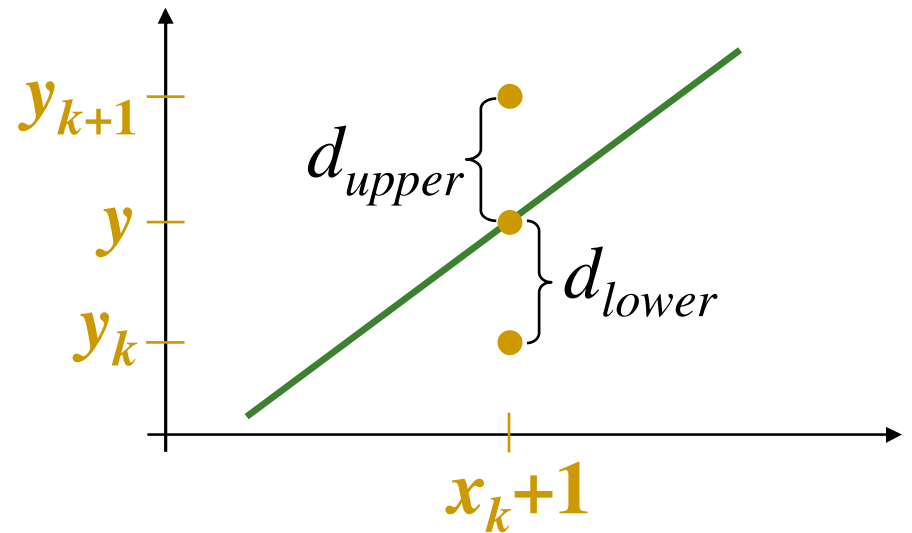
- избира се точката, която е **по-близо** до правата линия



Алгоритъмът на Брезенам

- За точка с координати x_k+1 дължината на вертикалните отсечки до математическата линия са означени с d_{upper} и d_{lower}
- Стойността на y координатата върху линията в точка с координатата x_k+1 е

$$y = m(x_k + 1) + b$$



Алгоритъмът на Брезенам

- Стойността на d_{lower} се изчислява като:

$$\begin{aligned}d_{lower} &= y - y_k \\ &= m(x_k + 1) + b - y_k\end{aligned}$$

- Стойността на d_{upper} се изчислява като:

$$\begin{aligned}d_{upper} &= (y_k + 1) - y \\ &= y_k + 1 - m(x_k + 1) - b\end{aligned}$$

- Решението кой пиксел се намира по-близо до математическата линия може да се вземе на базата на тези стойности

Алгоритъмът на Брезенам

- Най-простото решение се базира на разликата между позициите на двата пиксела

$$d_{lower} - d_{upper} = 2m(x_k + 1) - 2y_k + 2b - 1$$

- Ако означим с Δx и Δy разликите между координатите на двете крайни точки и заместим m с $\Delta y / \Delta x$:

$$\begin{aligned}\Delta x(d_{lower} - d_{upper}) &= \Delta x\left(2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1\right) \\ &= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + 2\Delta y + \Delta x(2b - 1) \\ &= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c\end{aligned}$$

Алгоритъмът на Брезенам

- Решаващият параметър p_k за k -тата стъпка по протежение на линията се определя като:

$$\begin{aligned} p_k &= \Delta x (d_{lower} - d_{upper}) \\ &= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c \end{aligned}$$

- **Знакът** на решаващия параметър съвпада със знака на разликата $d_{lower} - d_{upper}$
 - ако p_k е отрицателно, то се избира долния пиксел
 - в противен случай се избира горния пиксел

Алгоритъмът на Брезенам

- Тъй като изменението на координатите по абцисната ос x е със стъпка 1, изчисленията могат да са целочислени
- На стъпка $k+1$ решаващия параметър се определя като:

$$p_{k+1} = 2\Delta y \cdot x_{k+1} - 2\Delta x \cdot y_{k+1} + c$$

- Може да се определи разликата с решаващия параметър на предишната стъпка p_k :

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k)$$

Алгоритъмът на Брезенам

- Но x_{k+1} е всъщност равно на x_k+1 , следователно:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$$

- където $y_{k+1} - y_k$ има стойност или 0, или 1, в зависимост от знака на p_k

- На първата стъпка решаващият параметър p_0 се определя за началната точка от отсечката с координати (x_0, y_0) като:

$$p_0 = 2\Delta y - \Delta x$$

Алгоритъмът на Брезенам

Изчертаване на линия по алгоритъм на Брезенам (за $|m| < 1.0$)

1. Въвеждат се координатите на двете крайни точки, като лявата точка има координати (x_0, y_0)
2. Изчертава се точката (x_0, y_0)
3. Изчисляват се константите Δx , Δy , $2\Delta y$ и $(2\Delta y - 2\Delta x)$ и се определя началната стойност на решаващия параметър:
$$p_0 = 2\Delta y - \Delta x$$
4. За всяка стойност на x_k по протежение на линията (започвайки от $k = 0$) се определя:
 - ако $p_k < 0$ то следващата точка за изчертаване е (x_{k+1}, y_k) и: $p_{k+1} = p_k + 2\Delta y$
 - в противен случай следващата точка за изчертаване е (x_{k+1}, y_{k+1}) и: $p_{k+1} = p_k + 2\Delta y - 2\Delta x$
- 5. Стъпка 4 се повтаря $(\Delta x - 1)$ пъти

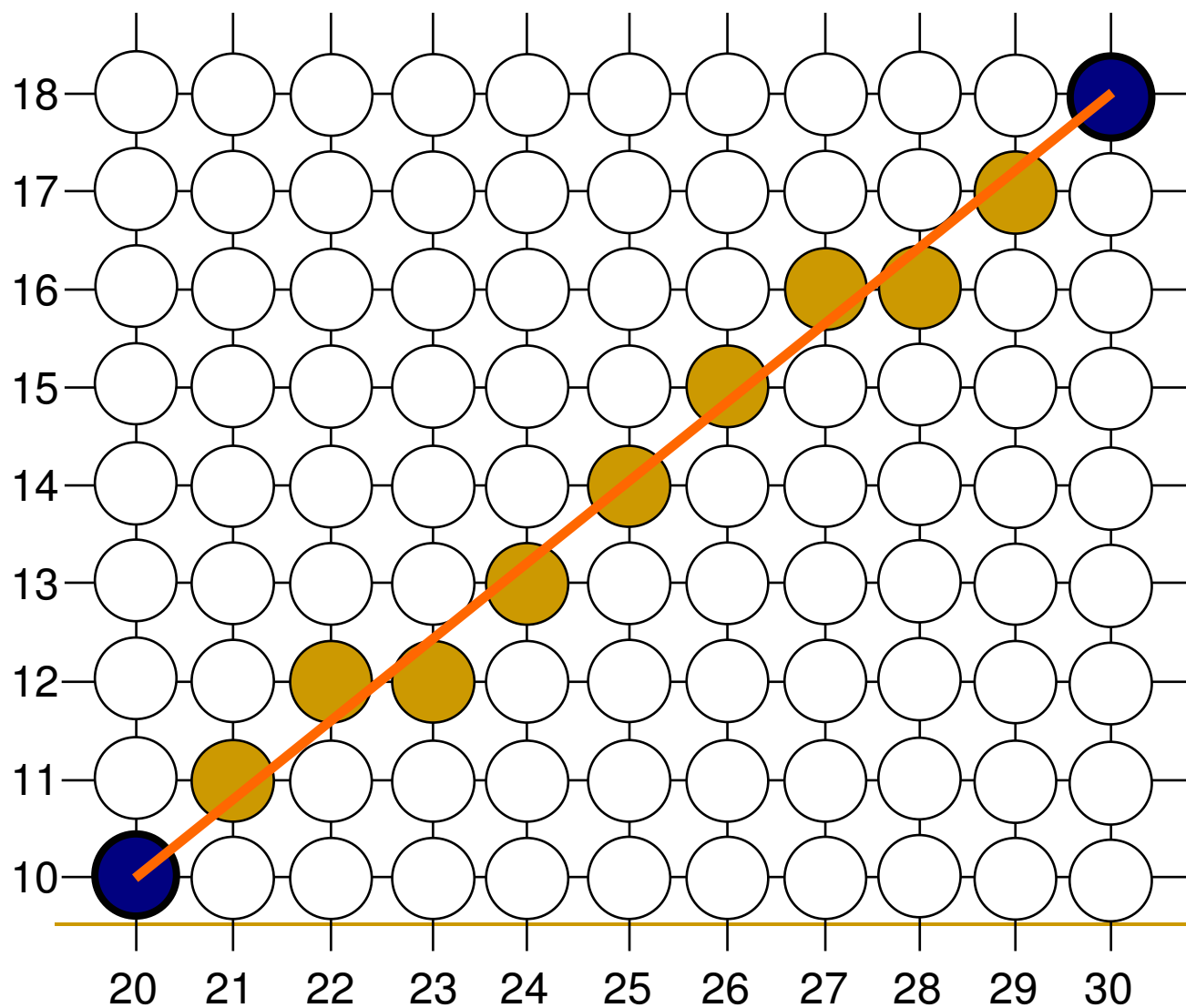
Алгоритъмът на Брезенам

```
void BresenhamLine (int x0, int y0, int x1, int y1) {
    int dx = x1 - x0;
    int dy = y1 - y0;
    int p = 2 * dy - dx;
    int incrE = 2 * dy;
    int incrNE = 2 * (dy - dx);
    int x = x0;
    int y = y0;
    writePixel(x, y);
    while (x < x1) {
        if (p <= 0) {           // East Case
            p = p + incrE;
        } else {               // Northeast Case
            p = p + incrNE;
            y++;
        }
        x++;
        writePixel(x, y);
    }
    /* while */
}
/* BresenhamLine */
```

Алгоритъмът на Брезенам

- Пример:
 - Да се изчертае линия от точка с координати (20, 10) до точка с координати (30, 18)
- *Стъпка 2:* Изчисляване на константите:
 - $\Delta x = 10$
 - $\Delta y = 8$
 - $2\Delta y = 16$
 - $2\Delta y - 2\Delta x = -4$
- *Стъпка 3:* Изчислява се началната стойност на решаващия параметър p_0 :
 - $p_0 = 2\Delta y - \Delta x = 6$

Алгоритъмът на Брезенам



k	p_k	(x_{k+1}, y_{k+1})
0	6	(21, 11)
1	2	(22, 12)
2	-2	(23, 12)
3	14	(24, 13)
4	10	(25, 14)
5	6	(26, 15)
6	2	(27, 16)
7	-2	(28, 16)
8	14	(29, 17)
9	10	(30, 18)

Изчертаване на линия

■ Алгоритъм DDA

□ недостатъци

- натрупването на грешки от закръгляване води до отклонение на растеризираната отсечка от реалната математическа линия
- операциите закръгляване и изчисленията с реални стойности са времеотнемащи

■ Алгоритъм на Брезенам

□ предимства

- бърз инкрементален алгоритъм
 - използва само целочислени изчисления
-

Изчертаване на окръжност

- Уравнение на окръжност в декартови координати

$$x^2 + y^2 = r^2$$

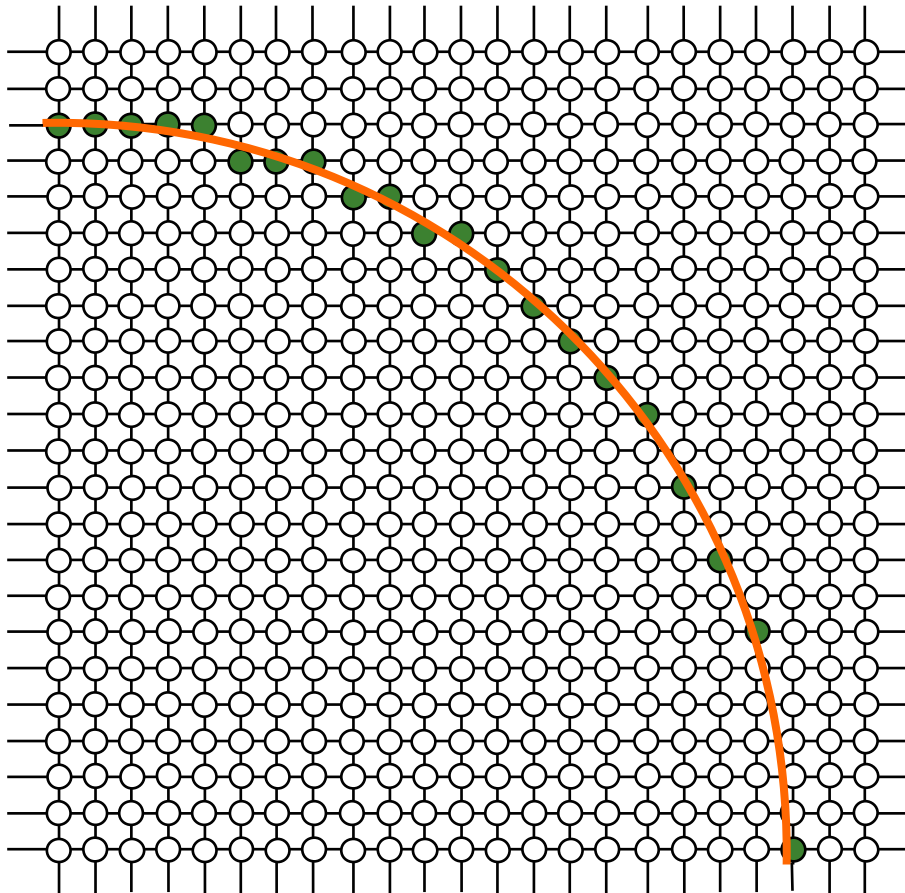
- където r е радиус на окръжността

- **Алгоритъм на грубата сила** за изчертаване на окръжност

- решава се уравнението на окръжността за y при единична стъпка на изменение на x :

$$y = \pm \sqrt{r^2 - x^2}$$

Алгоритъм на грубата сила



$$y_0 = \sqrt{20^2 - 0^2} \approx 20$$

$$y_1 = \sqrt{20^2 - 1^2} \approx 20$$

$$y_2 = \sqrt{20^2 - 2^2} \approx 20$$

⋮

$$y_{19} = \sqrt{20^2 - 19^2} \approx 6$$

$$y_{20} = \sqrt{20^2 - 20^2} \approx 0$$

Алгоритъм на грубата сила

- **Недостатъци** на алгоритъма на грубата сила
 - резултантната растеризирана окръжност съдържа много големи празнини (gaps)
 - особено в позициите, за които наклонът клони към вертикален
 - изчисленията не са много ефективни
 - операции за повдигане на квадрат (умножение)
 - операции за коренуване
 - желателно е да се избягват!

Изчертаване на окръжност

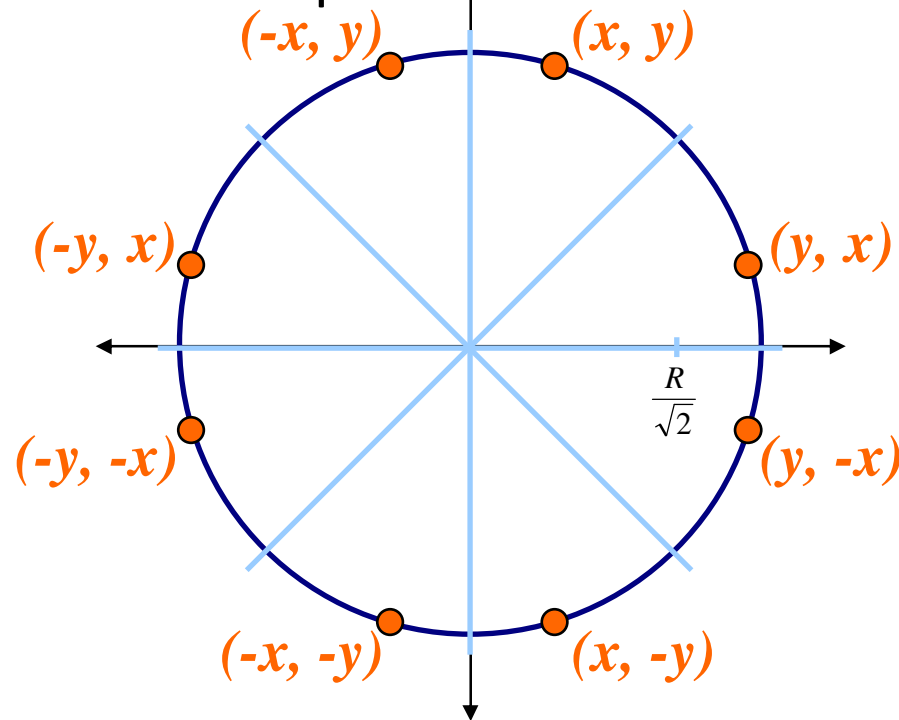
■ *Апроксимация на окръжност с отсечки*

- Изчертава се многоъгълник с върхове по окръжността
 - например стоъгълник
- Всяка отсечка се чертае от точка (X_1, Y_1) до (X_2, Y_2)
- Центърът на окръжността е в точка с координати (X_C, Y_C)
- В началото $(X_1, Y_1) = (X_C + R, Y_C)$
- На всяка стъпка
 - (X_2, Y_2) се изчислява по формулите
$$X_2 = X_C + R \cdot \cos\theta, Y_2 = Y_C + R \cdot \sin\theta$$
където $\theta \in [0, 2\pi)$ със стъпка $2\pi/100$
 - $(X_1, Y_1) = (X_2, Y_2)$

Изчертаване на окръжност

■ Симетрия на окръжността

- окръжност с център в точка с координати $(0, 0)$ има симетрия в осем посоки



- Ако се изчислят координатите (x, y) на една точка от окръжността, от нея могат да се получат още 7 точки за изчертаване
 - $(\pm x, \pm y)$ и $(\pm y, \pm x)$ също лежат върху окръжността
- Задачата се редуцира до определяне на пикселите от $1/8$ от окръжността

Изчертаване на окръжност

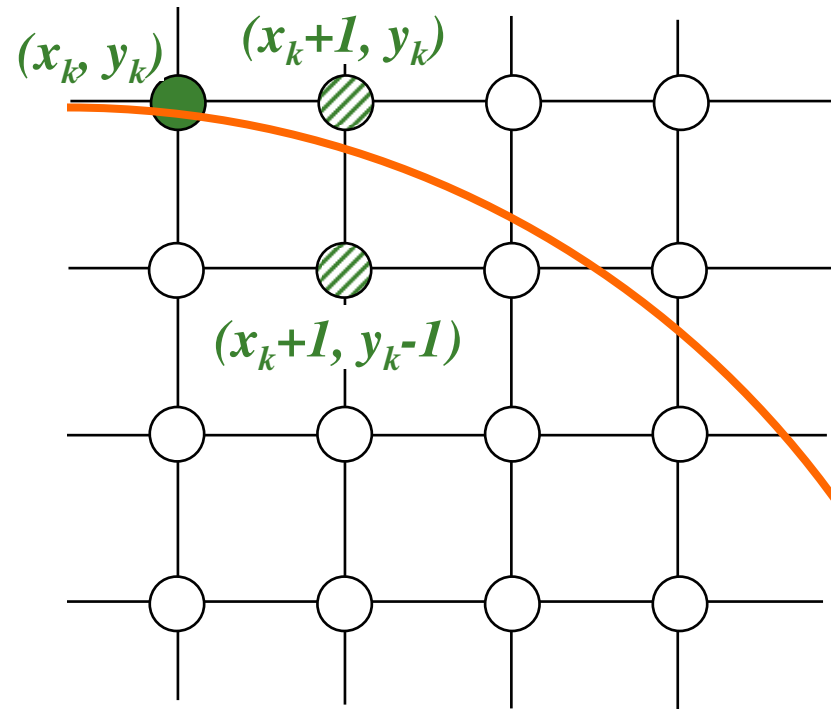
■ Алгоритъм на средната точка

- Mid-Point Circle Algorithm
- Jack Bresenham

■ Нека е изчертана точката с координати (x_k, y_k)

■ За следваща точка от окръжността трябва да се избере между (x_{k+1}, y_k) and (x_{k+1}, y_{k-1})

- избира се точката, която е **по-близо** до окръжността



Алгоритъм на средната точка

- Уравнението на окръжност може да се представи като

$$f_{circ}(x, y) = x^2 + y^2 - r^2$$

- За дадена точка с координати (x, y) е в сила:

$$f_{circ}(x, y) \begin{cases} < 0, & \text{ако } (x, y) \text{ е вътре в окръжността} \\ = 0, & \text{ако } (x, y) \text{ е върху окръжността} \\ > 0, & \text{ако } (x, y) \text{ е извън окръжността} \end{cases}$$

- *С оценяване на стойността на функцията $f_{circ}(x, y)$ за средните точки между кандидат-пикселите може да се определи кои пиксели да се изчертаят*

Алгоритъм на средната точка

- Нека последната изчертана точка е в пиксел с координати (x_k, y_k)
- Трябва да се избере следваща точка за изчертаване между (x_k+1, y_k) и (x_k+1, y_k-1)

- Решаващата променлива може да бъде определена като:

$$p_k = f_{circ}(x_k + 1, y_k - \frac{1}{2}) = (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2$$

- Ако $p_k < 0$ то средната точка е вътре в окръжността и пикселът с координата y_k е по-близо до окръжността
- В противен случай средната точка е извън окръжността и пикселът с координата y_k-1 е по-близо до окръжността

Алгоритъм на средната точка

- За постигане на по-голяма ефективност изчисленията се извършват инкрементално
- Разглежда се

$$p_{k+1} = f_{circ} \left(x_{k+1} + 1, y_{k+1} - \frac{1}{2} \right) = [(x_k + 1) + 1]^2 + \left(y_{k+1} - \frac{1}{2} \right)^2 - r^2$$

- или

$$p_{k+1} = p_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

където y_{k+1} е или y_k , или $y_k - 1$ в зависимост от знака на p_k

Алгоритъм на средната точка

- Началната стойност на решаващата променлива се определя като:

$$p_0 = f_{circ} \left(1, r - \frac{1}{2}\right) = 1 + \left(r - \frac{1}{2}\right)^2 - r^2 = \frac{5}{4} - r$$

- Ако $p_k < 0$ то следващата решаваща променлива е

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

- Ако $p_k > 0$ то следващата решаваща променлива е

$$p_{k+1} = p_k + 2x_{k+1} - 2y_k + 1$$

Алгоритъм на средната точка

Изчертаване на окръжност по алгоритъм на средната точка

1. Въвеждат се радиусът на окръжността r и координатите на центъра (x_c, y_c) и се определят координатите на първата точка от окръжност, центрирана спрямо началото на координатната система:

$$(x_0, y_0) = (0, r)$$

2. Изчислява се началната стойност на решаващия параметър:

$$p_0 = 5/4 - r$$

3. Прилага се следния тест за всяка стойност на x_k по протежение на линията (започвайки от $k = 0$):

- ако $p_k < 0$, то следващата точка за изчертаване от окръжността е (x_{k+1}, y_k) и се изчислява

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

- в противен случай следващата точка за изчертаване е (x_{k+1}, y_{k+1}) и новата стойност на решаващия параметър е

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

Алгоритъм на средната точка

*Изчертаване на окръжност по алгоритъм на средната точка
(продължение)*

4. Определят се симетричните точки в останалите 7 октанта
5. Изчертават се точки с координати, определени от изчислените (x, y) , отместени спрямо центъра на окръжността (x_c, y_c)
$$x = x + x_c \quad y = y + y_c$$
6. Стъпка 3 до 5 се повтарят докато $x \geq y$

Алгоритъм на средната точка

```
MEC (R) // 1/8th of a circle with radius R
{
    int x = 0, y = R;
    int delta_E, delta_SE;
    float decision;
    delta_E = 2*x + 3;
    delta_SE = 2(x-y) + 5;
    decision = (x+1)*(x+1) + (y + 0.5)*(y + 0.5) - R*R;
    Pixel(x, y);
    while( y > x ) {
        if (decision > 0) { /* Move east */
            decision += delta_E;
            delta_E += 2; delta_SE += 2; /*Update delta*/
        }
        else { /* Move SE */
            y--;
            decision += delta_SE;
            delta_E += 2; delta_SE += 4; /*Update delta*/
        }
        x++;
        Pixel(x, y);
    }
}
```

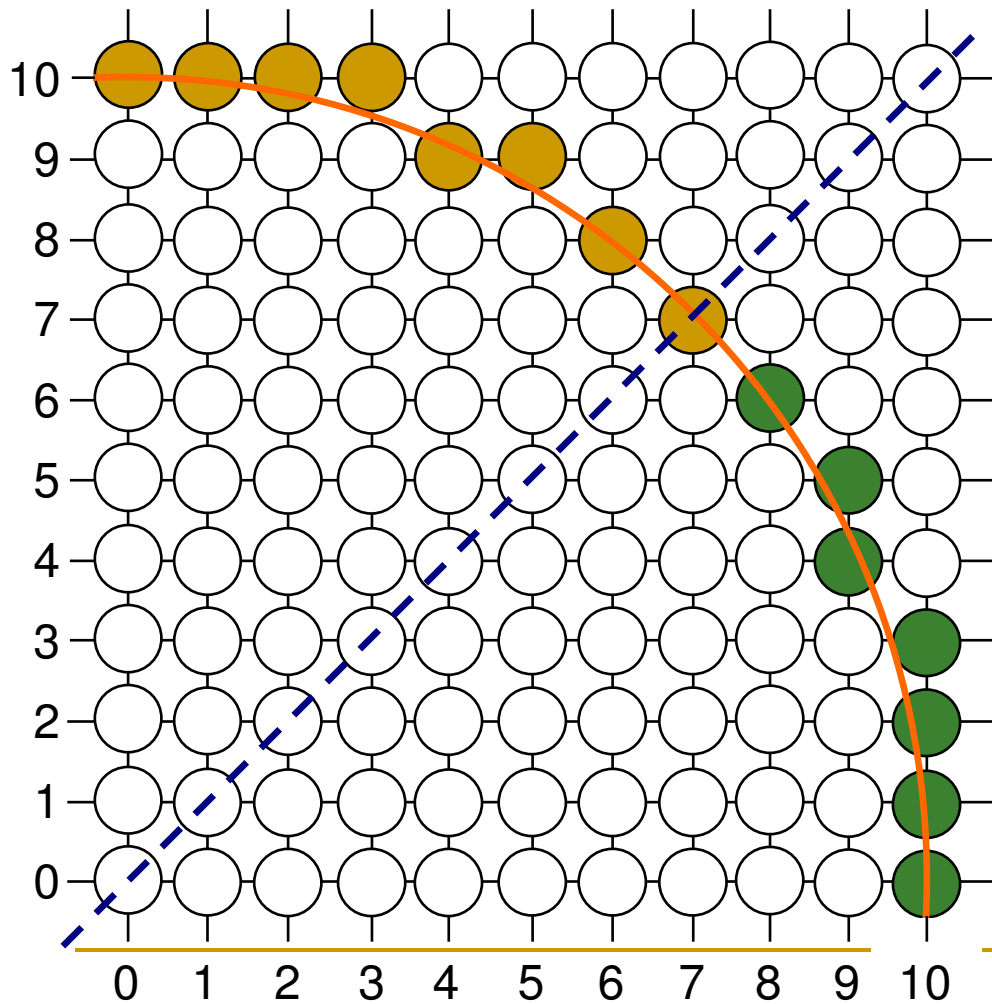
Алгоритъм на средната точка

- Пример
 - Да се изчертае окръжност с център с координати $(0, 0)$ и радиус 10

- *Стъпка 2:* Начален пиксел от окръжността (x_0, y_0)
 - $(x_0, y_0) = (0, 10)$

- *Стъпка 3:* Изчислява се началната стойност на решаващия параметър p_0 :
 - $p_0 = 5/4 - 10 = -8.75 \cong -9$

Алгоритъм на средната точка



k	p_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	(1, 10)	2	20
1	-6	(2, 10)	4	20
2	-1	(3, 10)	6	20
3	5	(4, 9)	8	18
4	-4	(5, 9)	10	18
5	7	(6, 8)	12	18
6	3	(7, 7)	6	6

Алгоритъм на средната точка

■ *Алгоритъм на средната точка*

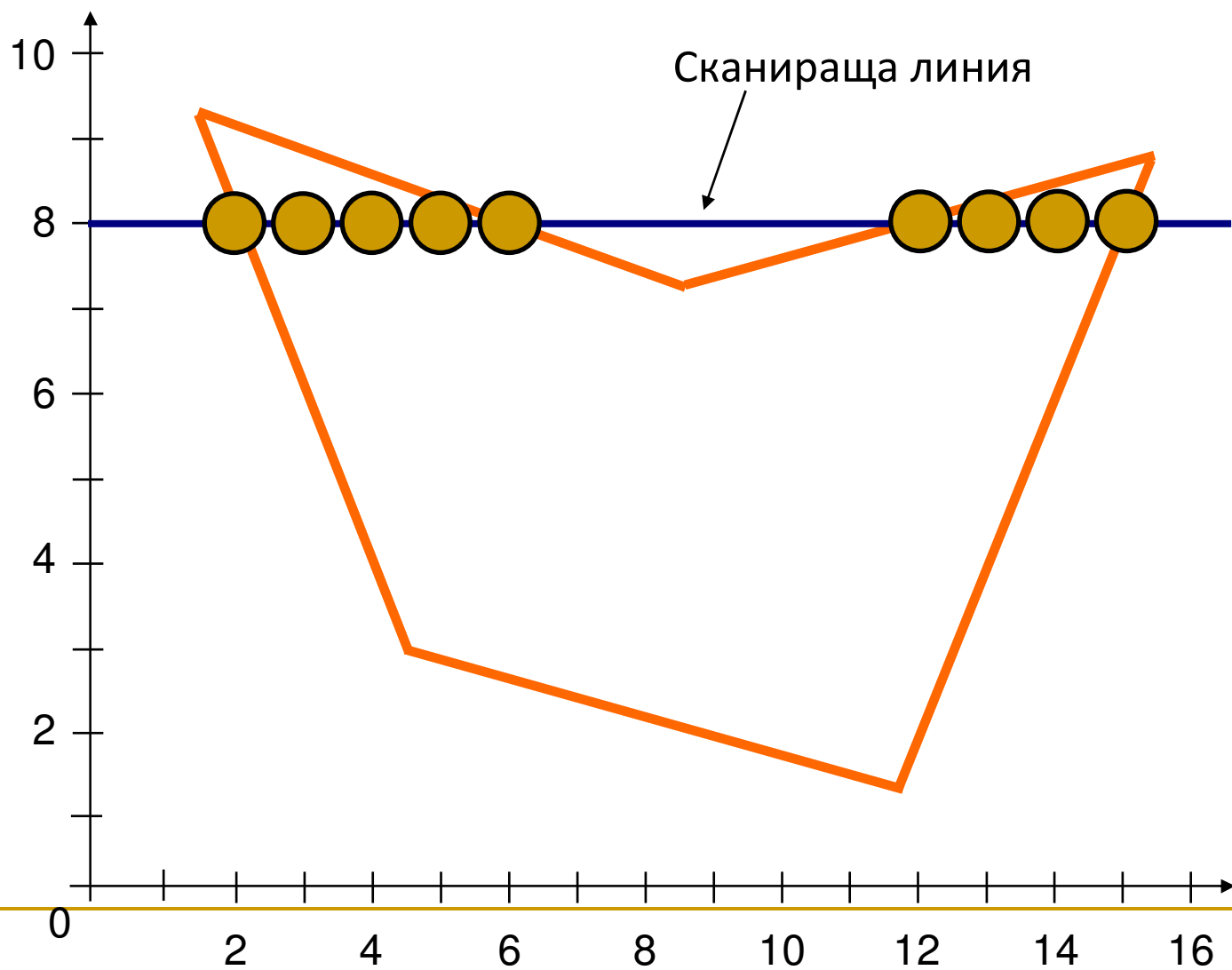
□ **предимства**

- използването на симетрия в осем посоки значително намалява изчисленията
- инкрементален алгоритъм
 - увеличаването на стойността на координатата по абцисната ос x със стъпка 1 изисква вземане на решение за следващата точка за изчертване измежду една от две възможни точки

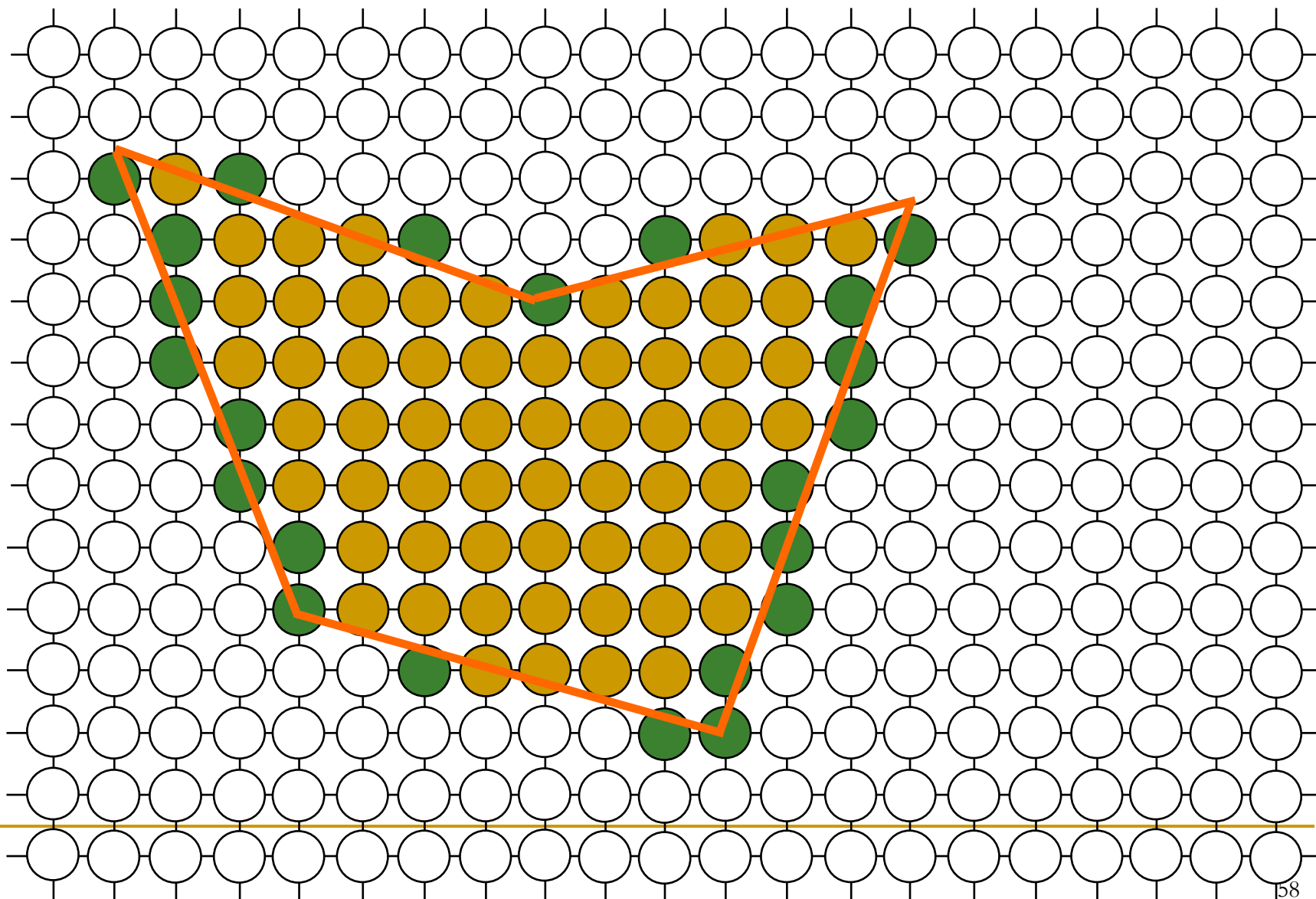
Запълване на геометрични фигури

- **Алгоритъм на сканиращата линия**
 - за изчертаване/запълване на полигони
- Определят се пресечните точки на сканиращата линия с всички ръбове на полигона
- Сортират се пресечните точки по стойност на нарастване на координатата x
- Запълват се всички пиксели, които са разположени между двойки пресечни точки и са вътре в полигона

Алгоритъм на сканиращата линия



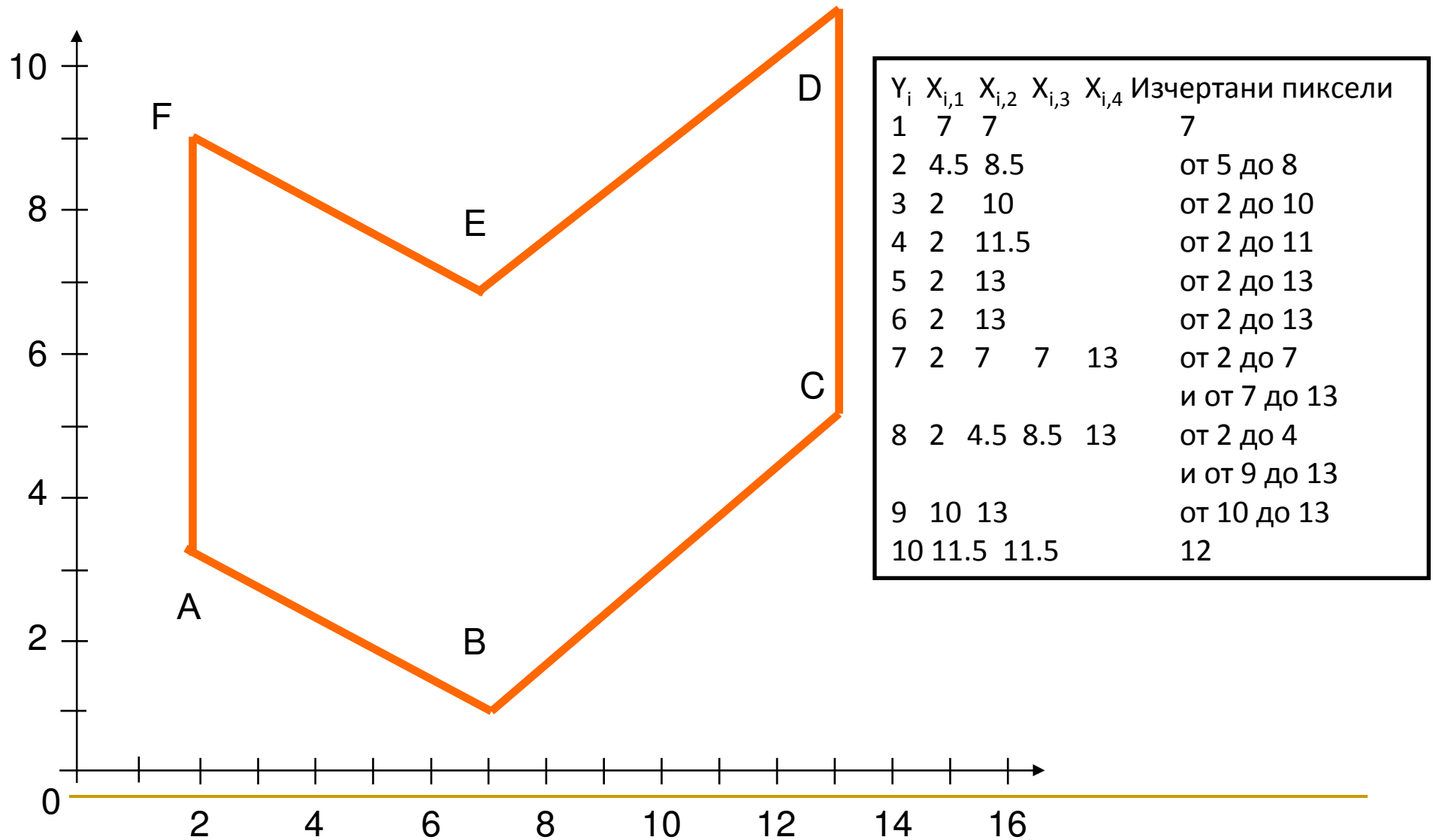
Алгоритъм на сканиращата линия



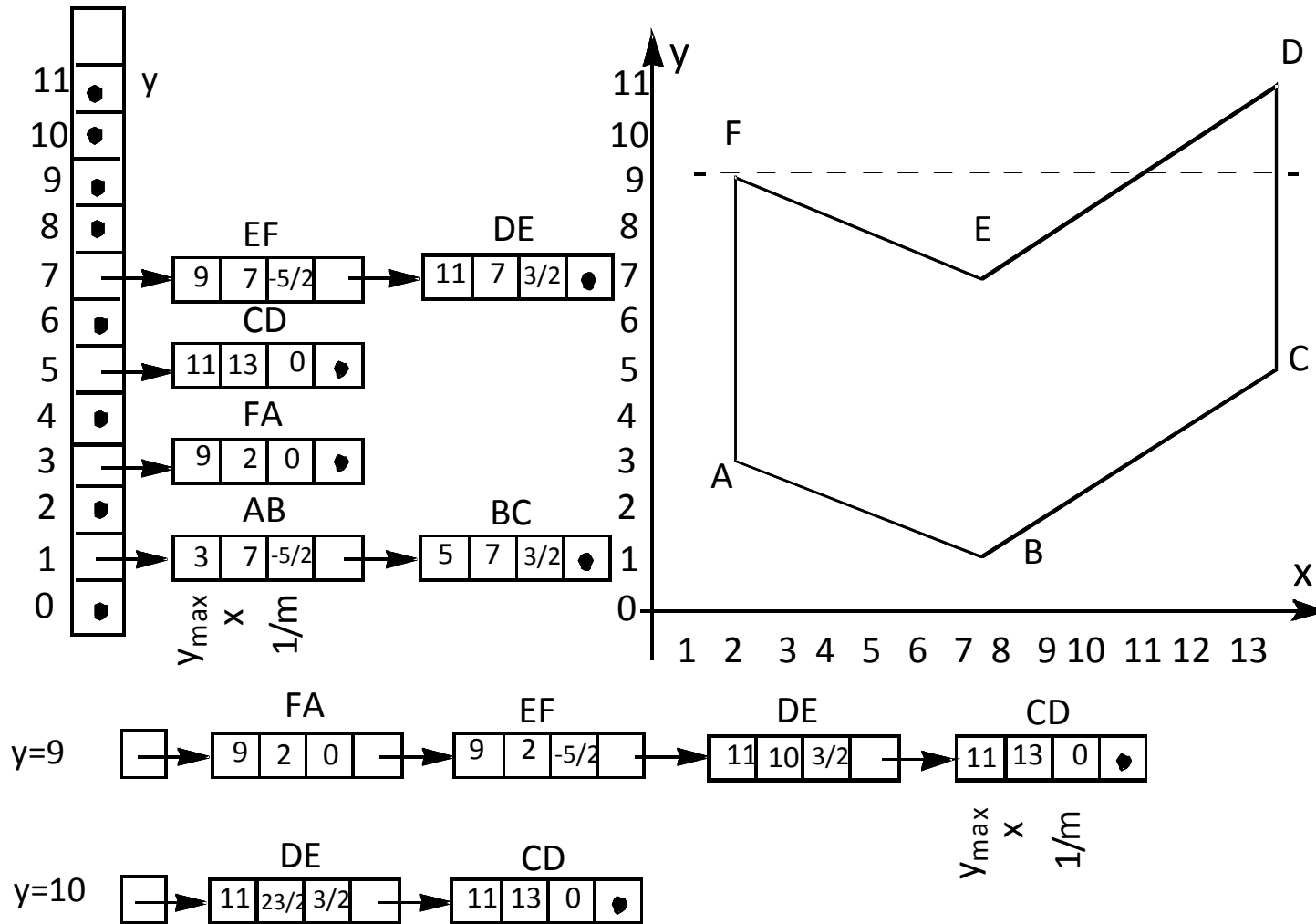
Алгоритъм на сканиращата линия

1. Намира се Y_{\max} и Y_{\min} за всички върхове на многоъгълника
2. Създава се подреден списък от сканиращи линии от Y_{\min} до $Y_{\max}-1$
3. За всеки ръб на многоъгълника се прави следната проверка:
 - Ако ръбът е хоризонтален, то се прескача
 - Изчислява се наклонът m на правата, определена от ръба и се изчислява стойността на $1/m$
 - Започва се от върха с по-малката координата Y_1 и за него в списъка за съответната сканираща линия се записва стойността на X координатата му
 - Y се увеличава с 1, изчислява се $X=X+1/m$ (ако $m \neq 0$, иначе X не се променя) и стойността на X се записва в списъка на X координатите за съответната сканираща линия, като се спазва подреждането на X координатите по големина
 - Продължава се до $Y_{\max}-1$ включително
4. За всяка сканираща линия от Y_{\min} до $Y_{\max}-1$ се изчертават всички пиксели за двойките X координати

Алгоритъм на сканиращата линия



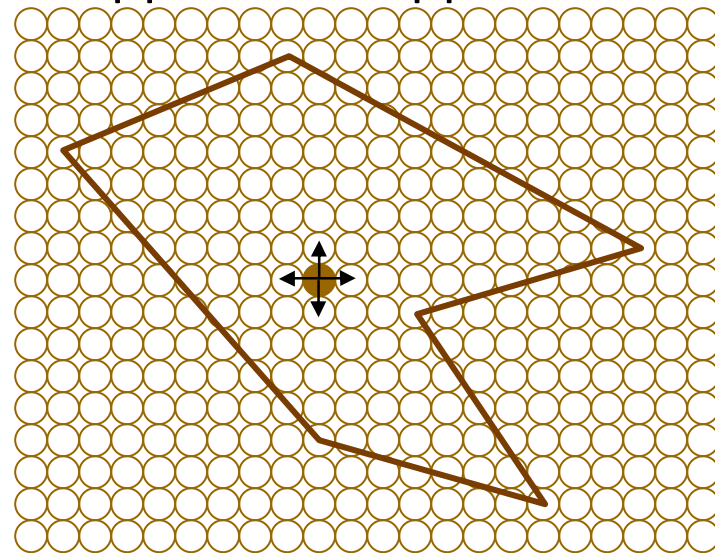
Алгоритъм на сканиращата линия



Запълване на геометрични фигури

■ “Наводняващ” алгоритъм

- избира се пиксел вътре в полигона
- изчертава се пиксела и съседните му пиксели
- продължава се рекурсивно докато се достигнат ръбовете на полигона

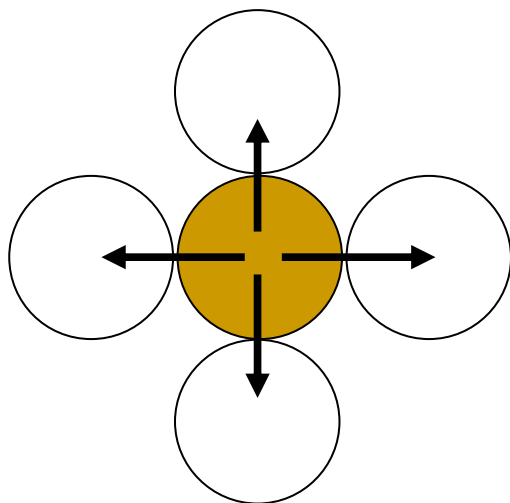


...

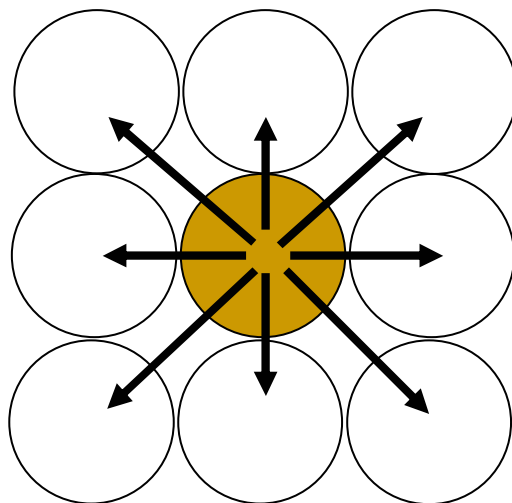
Запълване на геометрични фигури

■ “Наводняващ” алгоритъм

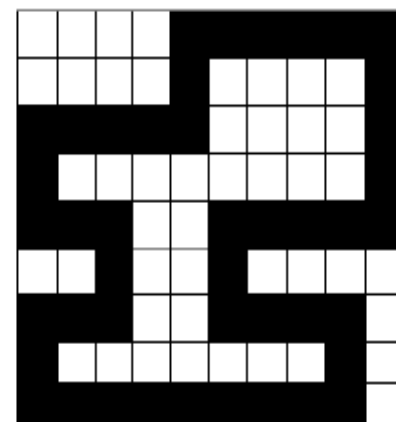
- съседни пиксели



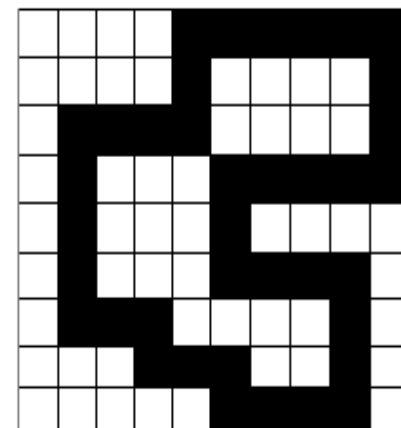
4 свързаност



8 свързаност



4-свързан полигон



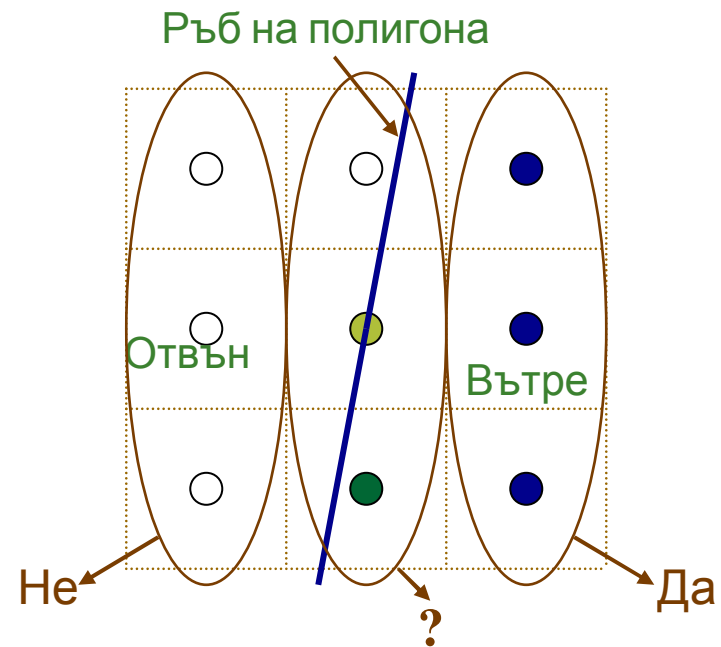
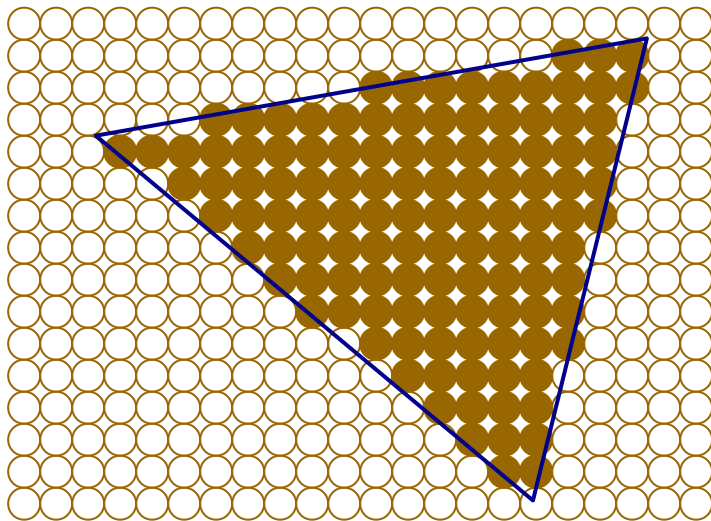
8-свързан полигон

...

Запълване на геометрични фигури

■ “Наводняващ” алгоритъм

- кои пиксели са вътре в границите на полигона?
 - тези, чиито център е вътре в полигона

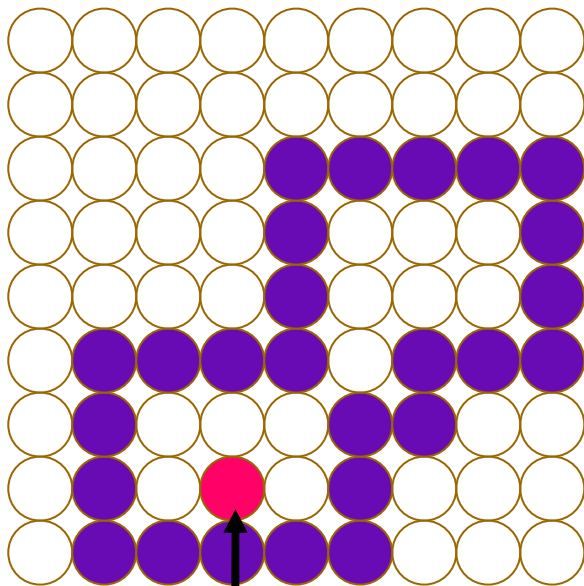


...

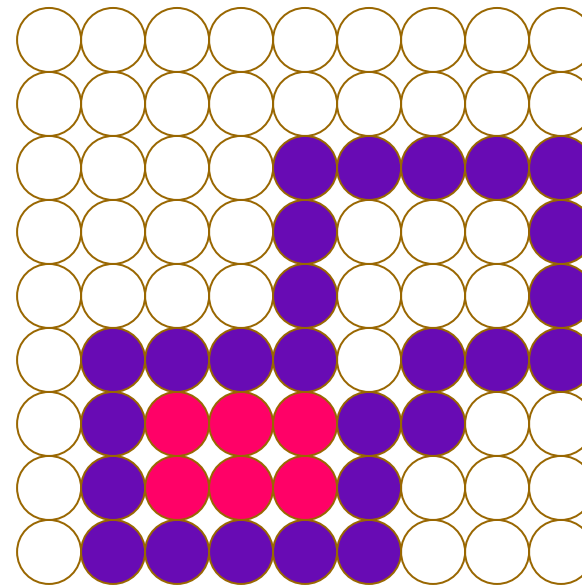
Запълване на геометрични фигури

■ “Наводняващ” алгоритъм

- пример – 4 свързаност



Начална точка



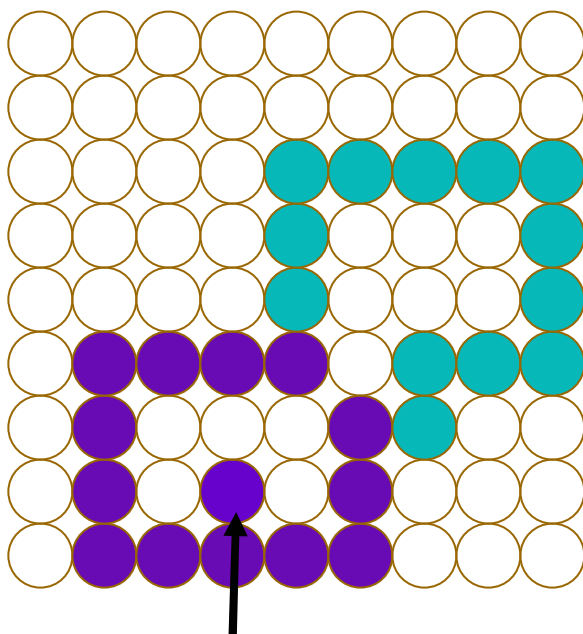
Запълнен полигон

...

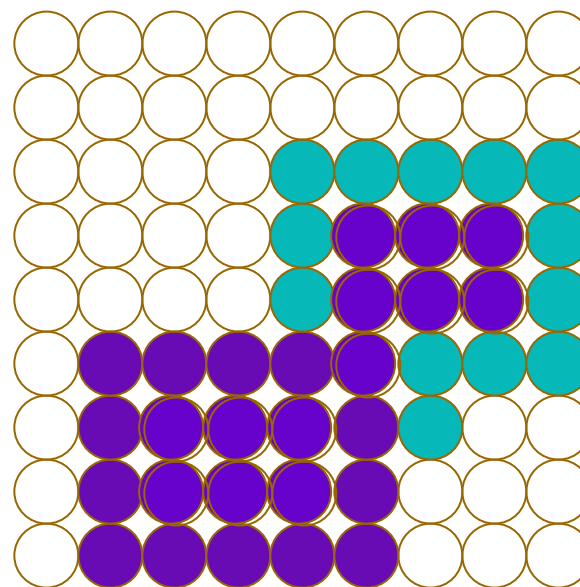
Запълване на геометрични фигури

■ “Наводняващ” алгоритъм

□ пример – 8 свързаност



Начална точка



Запълнен полигон

...

Запълване на геометрични фигури

■ *“Наводняващ” алгоритъм*

□ **проблеми**

- рекурсивност
- резултатът зависи от свързаността
 - може да се “премине” в съседен полигон
 - може да не се запълни правилно полигон

КРАЙ

Следваща тема:
3D трансформации