Лекц:
Bresenham шулуун Зурах алгоритм, Тойрог зурах & Polygon дүүргэх

Агуулга

Өнөөдрийн хичээлээр бид дараах зүйлийг судлах болно:

- Bresenham-ийн шулууныг зурах алгоритм
- Шулууныг зурах алгоритмуудын харьцуулалт
- Тойрог зурах алгоритм
  - Энгийн арга
  - mid-point тойрог алгоритм
- Олон өнцөгт (Polygon) дүүгэх алгоритм
- Растер зургийн алгоритмын тойм

Bresenham Line Algorithm

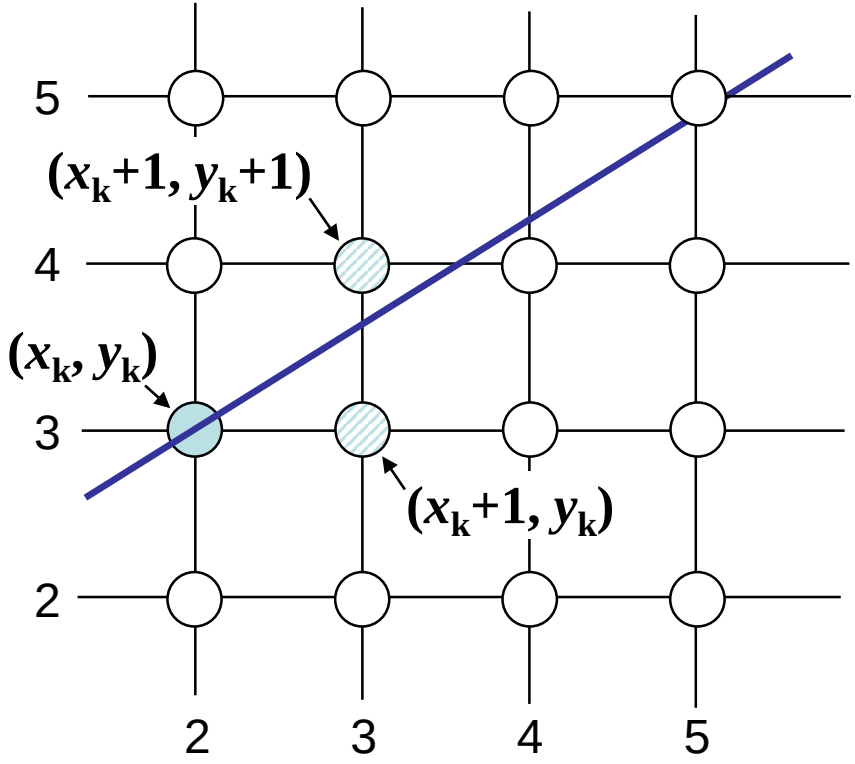
Bresenham алгоритм нь өсөн нэмэгдэх scan conversion алгоритм юм.

Энэ алгоритмын хамгийн том давуу тал нь зөвхөн бүхэл тооны тооцоолол ашигладагт оршино.

Жак Брезенхэм академид орохоосоо өмнө IBM-д 27 жил ажилласан. Брезенхам өөрийн алдартай алгоритмуудаа 1960-аад оны эхээр IBM-д ажиллаж байхдаа боловсруулсан.

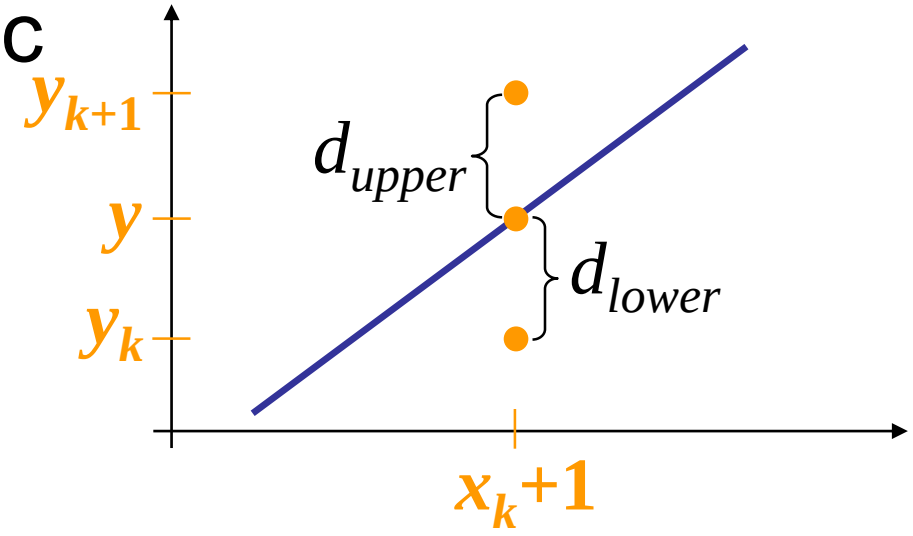
Гол санаа нь

X тэнхлэгийн дагуу нэгж интервалаар шилжиж, алхам бүрт хоёр өөр у координатын хооронд сонгоно



Bresenham шулууны алгоритмыг гаргаж авах нь

Математик шугамнаас босоо тусгаарлалтыг жишээ байрлал x_k+1 d_{upper} ба d_{lower} гэж тэмдгэлсэн



x_k+1 дээрх математик шулуун дээрх y координат:

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

Bresenham шулууны алгоритмыг гаргаж авах нь

Иймд d_{upper} ба d_{lower} дараах байдлаар өгөгдсөн:

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

ба:

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

Бид эдгээрийг ашиглан аль пикселийг математикийн шугаманд ойртуулах талаар энгийн шийдвэр гаргах боломжтой

Bresenham шулууны алгоритмыг гаргаж авах нь (cont…)

Энэхүү энгийн шийдвэр нь хоёр пикселийн байршлын зөрүү дээр суурилсан болно:

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

m –г $\Delta y/\Delta x$ орлуулъя. Δx ба Δy нь end-points хоорондын зөрүү юм:

$$\Delta x(d_{lower} - d_{upper}) = \Delta x(2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1)$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + 2\Delta y + \Delta x(2b - 1)$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

Bresenham шулууны алгоритмыг гаргаж авах нь (cont…)

Иймд шулууны дагуух k дахь алхамын шийдвэрийн параметр p_k дараах байдлаар өгөгдөнө

$$p_k = \Delta x(d_{lower} - d_{upper})$$

$$= 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

Шийвдэрийн параметр p_k нь $d_{lower} - d_{upper}$ ижил. Хэрэв p_k сөрөг гарвал доод пикселийг эсрэг тохиолдолд дээд пикселийг сонгоно.

Bresenham шулууны алгоритмыг гаргаж авах нь (cont…)

k+1 алхам дээрх шийдвэрийн параметр:

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

Үүнээс p_k хасвал дараах тэгшитгэл болно:

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

Bresenham шулууны алгоритмыг гаргаж авах нь (cont…)

Гэвч x_{k+1} нь x_k+1 ижил тул:

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

y_{k+1} - y_k нь p_k тэмдгээс хамаарч 0 эсвэл 1 байна. Анхны шийдвэрийн параметр p0 нь (x0, y0) дээр тооцож авч үзсэнийг дараах байдлаар өгнө:

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

Bresenham Line Algorithm

BRESENHAM-ийн шулуун зурах алгоритм
(for $|m| < 1.0$)

1. Оролтонд 2 төгсөглийн цэг авч left end-point –г (x_0, y_0) хадгална
2. point (x_0, y_0) тэмдэглэнэ.
3. Δx, Δy, $2\Delta y$, болон $(2\Delta y - 2\Delta x)$ тогтмолуудыг тооцоолж, шийдвэрийн параметрийн эхний утгыг авна:
$$p_0 = 2\Delta y - \Delta x$$
4. Шулуун дээрх x_k бүрт $k = 0$ эхлэх ба дараах шалгалтыг хийнэ. Хэрэв $p_k < 0$ *бол* дараагийн тэмдэглэх цэг (x_k+1, y_k) :
$$p_{k+1} = p_k + 2\Delta y$$

The Bresenham Line Algorithm (cont…)

Эсрэг тохиолдолд дараагийн цэг тэмдэглэх (x_k+1, y_k+1) ба:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

5. 4-р алхамыг (Δx – 1) удаа давтана.

Bresenham жишээ

(20, 10) -аас (30, 18) хүртэлх шулууныг зурцгаая

Эхлээд бүх тогтмолуудыг тооцоолно:

Δx: 10

Δy: 8

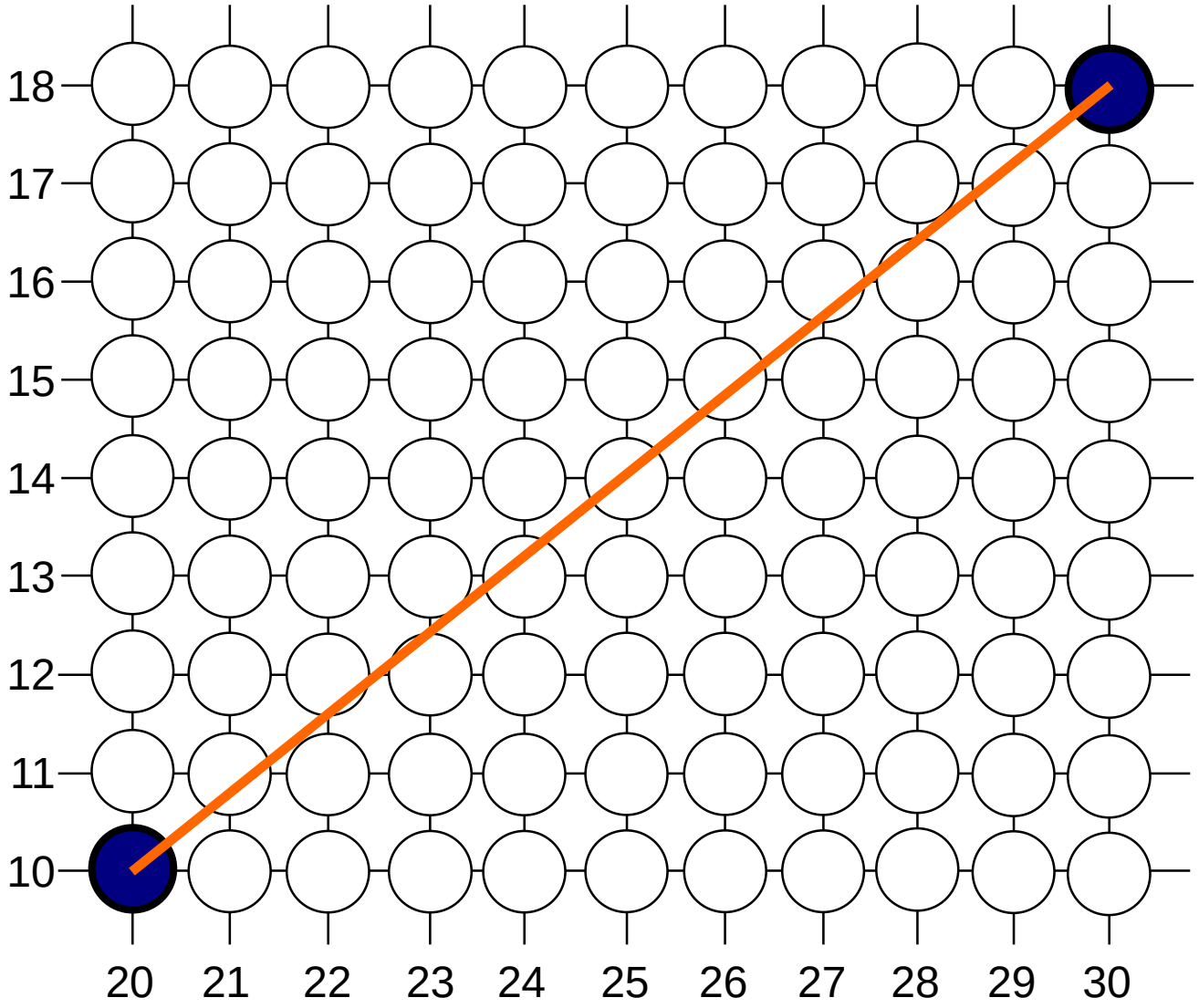
$2\Delta y$: 16

$2\Delta y - 2\Delta x$: -4

Анхдагч шийдвэрийн параметр p_0:

$p0 = 2\Delta y - \Delta x$ = 6

Bresenham жишээ (үргэлжлэл…)

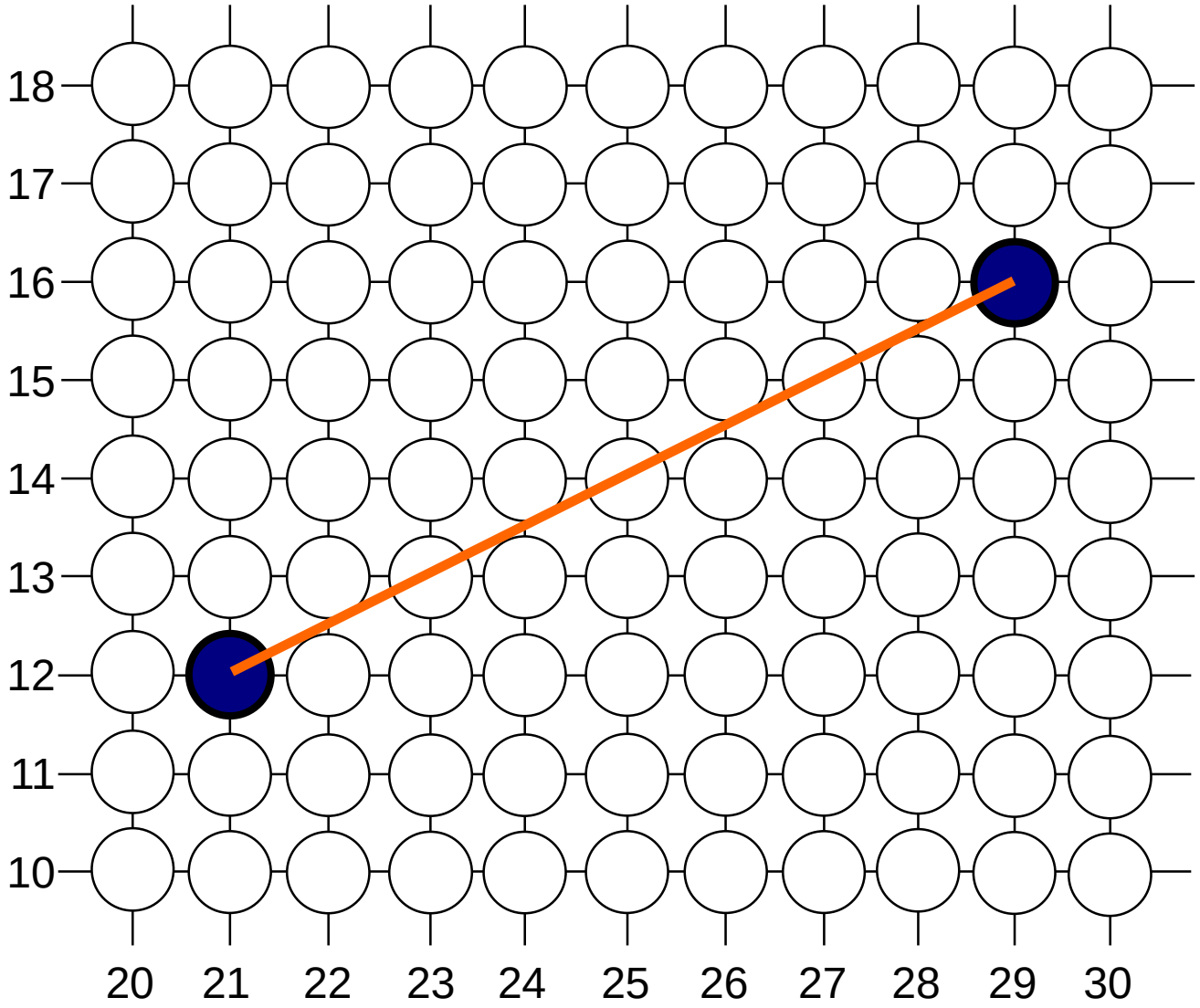


k	p_k	(x_{k+1},y_{k+1})
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		

Bresenham Дасгал ажил

(21,12) -с (29,16) хоорондох шулууны хувьд Bresenham шулуун зурах алгоритмын үе шатуудаар дамжина уу

Bresenham Дасгал ажил(cont…)



k	p_k	(x_{k+1},y_{k+1})
0		
1		
2		
3		
4		
5		
6		
7		
8		

Bresenham шулууны алгоритмын дүгнэлт

Bresenham шулууны алгоритм дараах давуу талуудтай:

- Хурдтай өсөх алгоритм
- Зөвхөн бүхэл тооны тооцооллыг ашигладаг

DDA алгоритмтай харьцуулахад DDA дараах асуудлууд бий:

- Ойролцоолсон алдааны хуримтлал нь пикселжүүлсэн шугамыг төлөвлөснөөс холдуулдаг.
- Ойролцоолох үйлдлүүд ба хөвөгч таслалтай цэгийн арифметик нь цаг хугацаа их шаарддаг.

Digital differential algoritm(DDA)

```
#define Round(a) ((int)(a+0.5)
void lineDDA(int xa, int ya, int xb, int yb)
{ int dx=xb-xa, dy=yb-ya, steps, k;
     float xInc, yInc, x=xa, y=yb;
     If(abs(dx)>abs(dy)) steps=abs(dx);
      else steps=abs(dy);
   xInc=dx/(float) steps;
   yInc=dy/(float) steps;
   SetPixel(Round(x),Round(y));
   for(k=o;k<steps;k++){
                              x+=xInc, y+=yInc;
                              setPixel(Round(x),Round(y));
                             }
   }
```

Тойрог зурах энгийн алгоритм

Тойргийн томъёо нь:

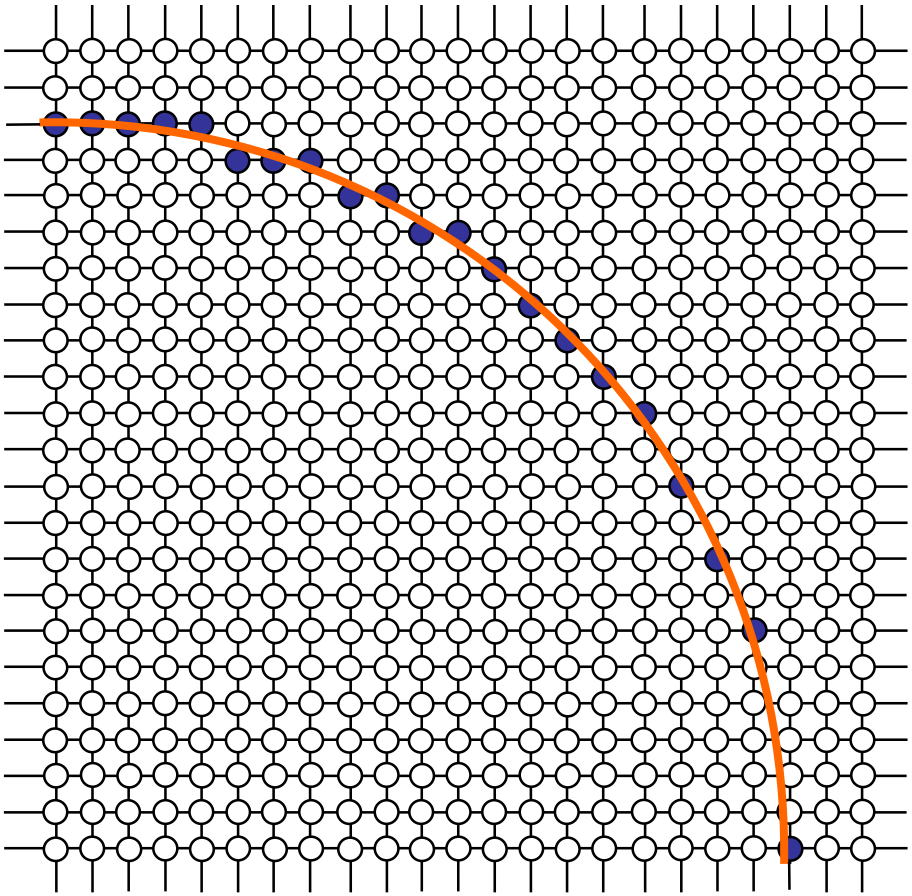
$$x^2 + y^2 = r^2$$

r нь тойргийн радиус

Иймд бид нэгж x интервал дээрх y томъёог шийдэх замаар энгийн тойрог зурах алгоритмыг бичих боломжтой:

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

Тойрог зурах энгийн алгоритм(cont…)



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

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

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

$$\vdots$$

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

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

Тойрог зурах энгийн алгоритм (cont…)

Энэ нь тийм ч сайн шийдэл биш юм!

Нэгдүгээр, тойргийн үр дүн налуу нь босоо тэнхлэгт ойртох том зайтай байдаг

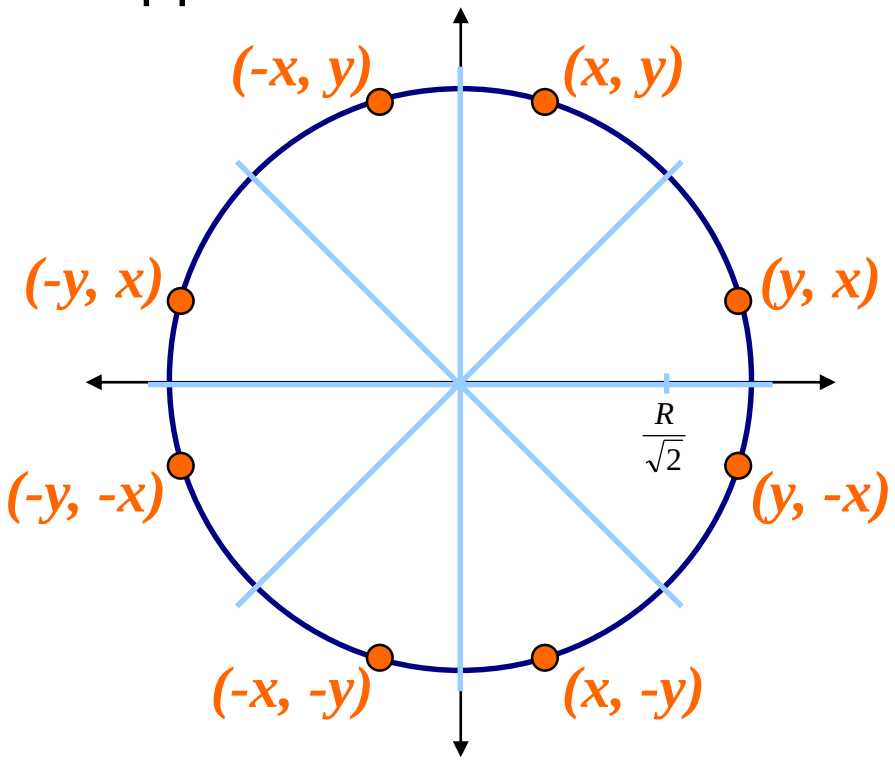
Хоёрдугаарт, тооцоолол тийм ч үр дүнтэй биш байна

- квадрат (үржүүлэх) үйлдлүүд
- Квадрат язгуур авах үйлдлүүд– эдгээрээс зайлсхийх хэрэгтэй!

Бид илүү үр дүнтэй, илүү нарийвчлалтай шийдэл хэрэгтэй

Найм-талт тэгш хэм

Тойрог зурах алгоритмаа илүү үр дүнтэй болгохын тулд бидний анзаарч байгаа хамгийн эхний зүйл бол (0, 0) төвтэй тойрог нь найман талын тэгш хэмтэй байх явдал юм.



Дундаж цэгийн тойргийн алгоритм (Mid-Point Circle Algorithm)

Шулууны тохиолдлууд ижил нөхцөлд тойрог зурах өсөлтийн алгоритм бий – *mid-point circle algorithm*

mid-point circle algorithm хувьд найман-талт тэгш хэмийг ашиглах тул зөвхөн тойргийн баруун дээд цэгийг тооцоолох ба дараа нь тэгш хэмийг ашиглан үлдсэн цэгүүдийг авна.

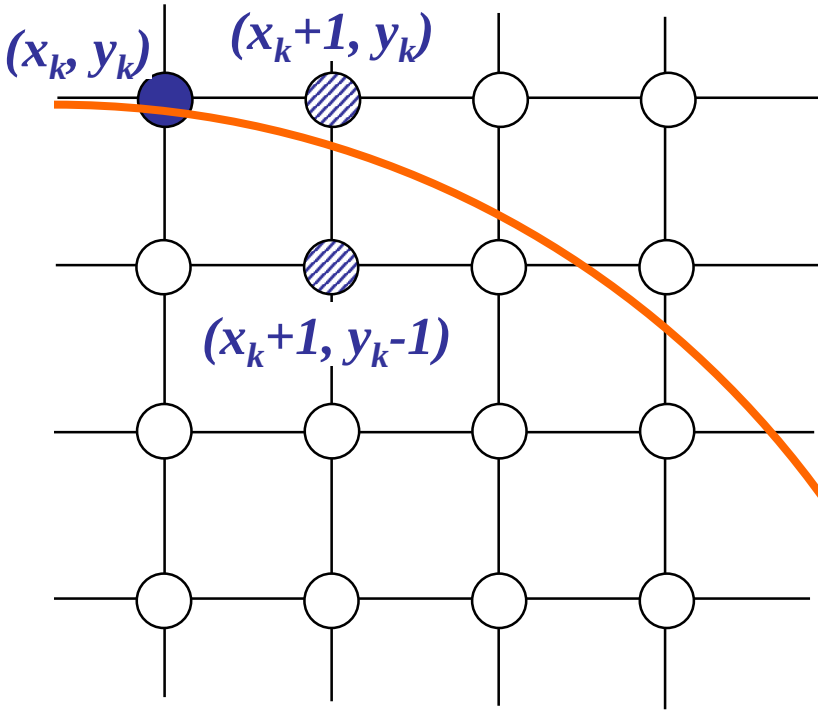
mid-point circle algorith-ийг бидний өмнө сонсож байсан Jack Bresenham хөгжүүлсэн.

Дундаж цэгийн тойргийн алгоритм (cont…)

(x_k, y_k) цэгийг тэмдгэлсэн гэж үзье.

Дараагийн цэг бол (x_k+1, y_k) ба (x_k+1, y_k-1) хоорондох сонголт юм.

Бид тойрогтой хамгийн ойрхон цэгийг сонгоно. Бид энэ сонголтыг хэрхэн хийх вэ?



Дундаж цэгийн тойргийн алгоритм (cont…)

Бидэнд өгөгдсөн тойргийн тэгшитгэлийг бага зэрэг давтъя:

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

Тэгшитгэлийг дараах байдлаар үнэлнэ:

$$f_{circ}(x, y) \begin{cases} < 0, \text{ if } (x, y) \text{ is inside the circle boundary} \\ = 0, \text{ if } (x, y) \text{ is on the circle boundary} \\ > 0, \text{ if } (x, y) \text{ is outside the circle boundary} \end{cases}$$

Оролцогч цэгүүдийн хоорондох дундаж цэг дээр энэ функцийг тооцоолсоноор бид шийдвэрээ гаргана.

Дундаж цэгийн тойргийн алгоритм (cont…)

Бид (x_k,y_k) дээрх цэгийг тэмдгэлсэн гэж үзвэл (x_k+1,y_k) ба (x_k+1,y_k-1) хоорондох сонголтыг хийнэ.

Бидний шийдвэрийн хувьсагч:

$$p_k = f_{circ}(x_k+1, y_k - 1/2)$$

$$= (x_k+1)^2 + (y_k - 1/2)^2 - r^2$$

$p_k < 0$ бол дундаж цэг тойргийн дотор орших ба y_k нь тойрогтой ойр оршино.

Эсэргээрээ дундаж цэг гадна оршиж байвал y_k-1 ойр байна

Дундаж цэгийн тойргийн алгоритм (cont…)

Аливаа зүйлийг аль болох үр дүнтэй байлгахын тулд бүх тооцоогоо алхам алхмаар хийнэ

Эхлээд:

$$p_{k+1} = f_{circ}\left(x_{k+1}+1, y_{k+1}-\tfrac{1}{2}\right)$$

$$= [(x_k+1)+1]^2 + \left(y_{k+1}-\tfrac{1}{2}\right)^2 - r^2$$

буюу:

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

y_{k+1} нь p_k хамаарч y_k эсвэл y_k-1 байна.

Дундаж цэгийн тойргийн алгоритм (cont…)

Эхний шийдвэрийн хувьсагч дараах байдлаар өгөгдсөн:

$$p_0 = f_{circ}(1, r - 1/2)$$

$$= 1 + (r - 1/2)^2 - r^2$$

$$= 5/4 - r$$

Хэрэв $p_k < 0$ бол дараагийн шийдвэрийн хувьсагч:

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

$p_k > 0$ бол шийдвэрийн хувьсагч:

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

Дундаж цэгийн тойргийн алгоритм

MID-POINT CIRCLE ALGORITHM

- Input radius r and circle centre (x_c, y_c), then set the coordinates for the first point on the circumference of a circle centred on the origin as:

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

- Calculate the initial value of the decision parameter as:

$$p_0 = 5/4 - r$$

- Starting with $k = 0$ at each position x_k, perform the following test. If $p_k < 0$, the next point along the circle centred on $(0, 0)$ is (x_k+1, y_k) and:

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

Дундаж цэгийн тойргийн алгоритм (cont…)

Otherwise the next point along the circle is (x_k+1, y_k-1) and:

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

4. Determine symmetry points in the other seven octants
5. Move each calculated pixel position (x, y) onto the circular path centred at (x_c, y_c) to plot the coordinate values:

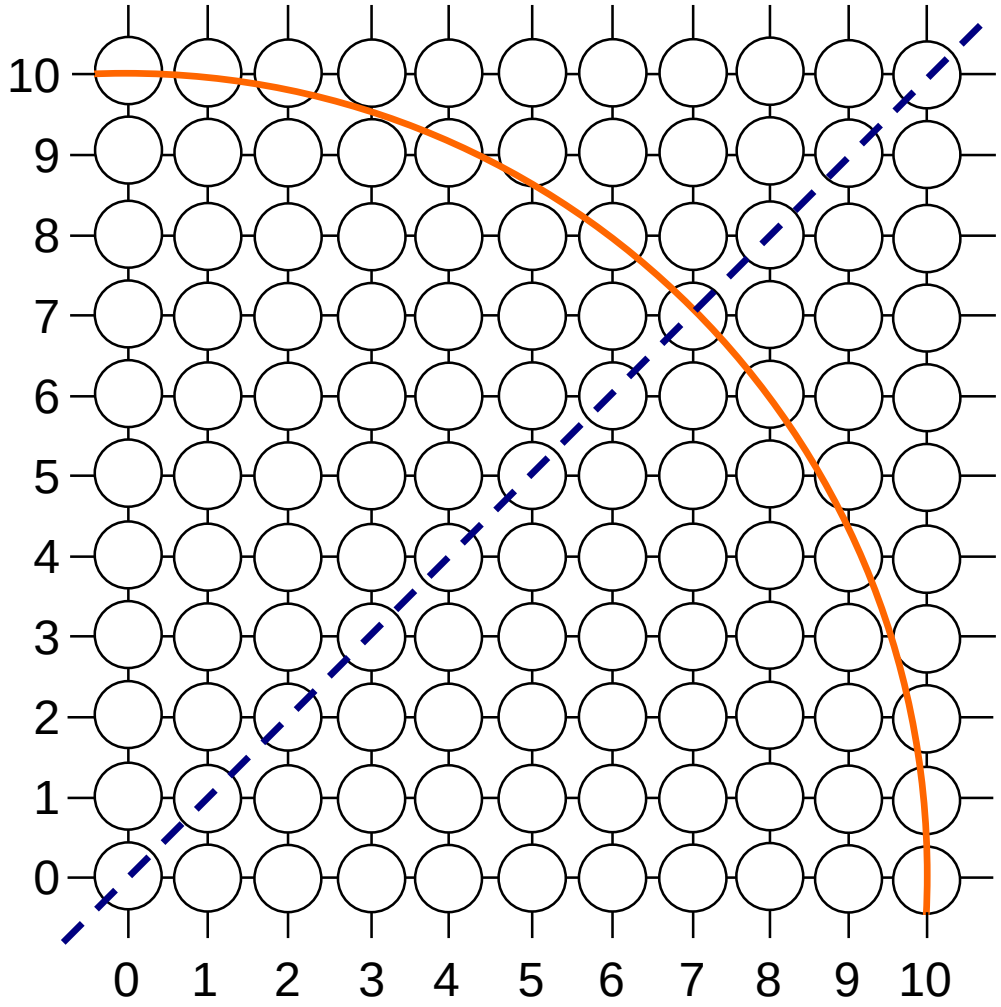
$$x = x + x_c \qquad y = y + y_c$$

6. Repeat steps 3 to 5 until $x >= y$

Дундаж цэгийн тойргийн алгоритмын жишээ

Дундаж цэгийн тойргийн алгоритмыг үйлдлээр нь харахын тулд (0,0) төвтэй, 10 радиустай тойрог зурахад ашиглана уу.

Дундаж цэгийн тойргийн алгоритмын жишээ(cont…)

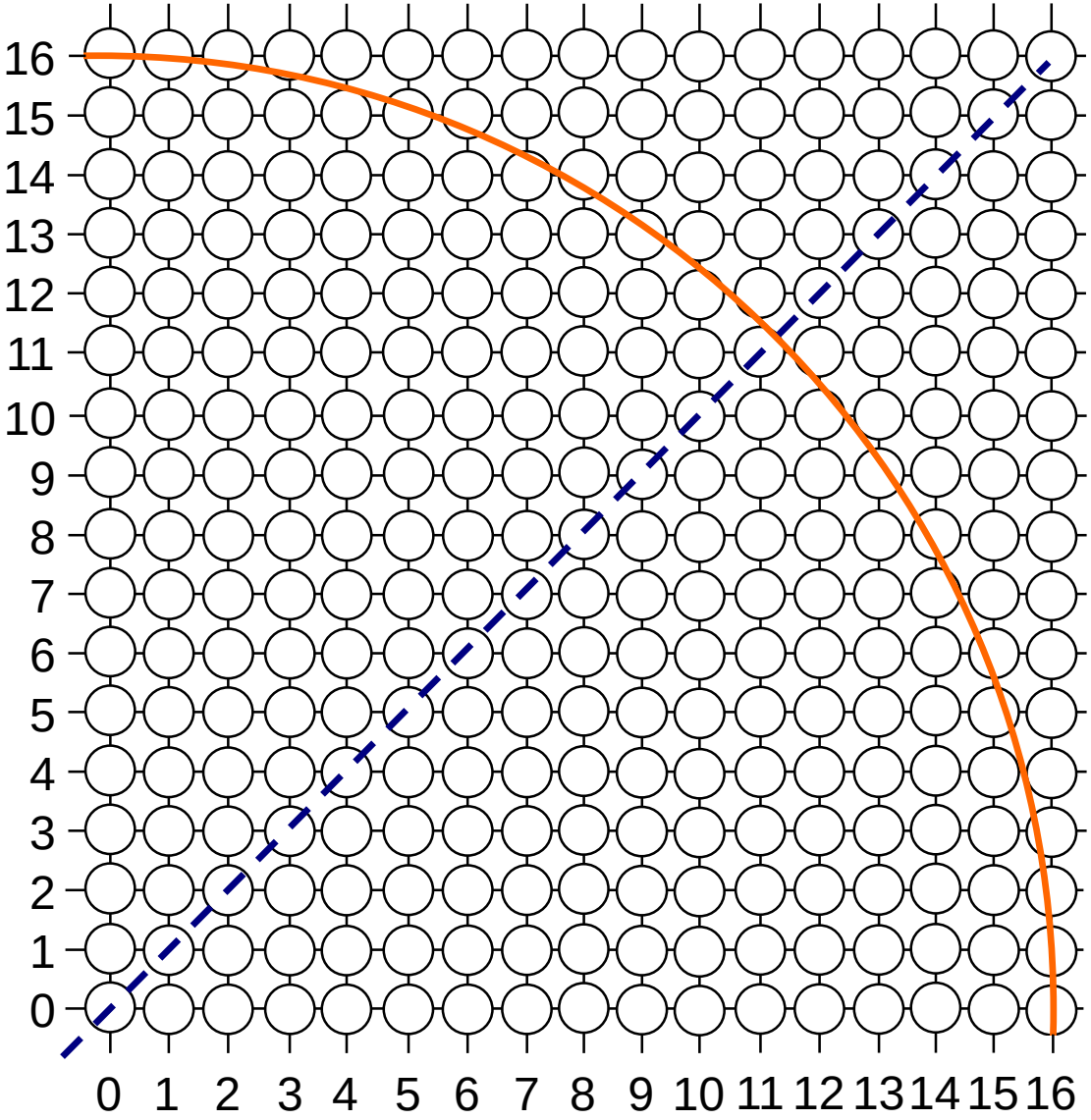


k	p_k	(x_{k+1},y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0				
1				
2				
3				
4				
5				
6				

Дундаж цэгийн тойргийн алгоритмын дасгал ажил

Дундаж цэгийн тойргийн алгоритмийг ашиглан 15 радиустай (0,0) төвтэй тойрог зур.

Mid-Point Circle Algorithm Example (cont…)



k	p_k	(x_{k+1},y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				

Mid-Point Circle Algorithm Summary

The key insights in the mid-point circle algorithm are:

- Eight-way symmetry can hugely reduce the work in drawing a circle
- Moving in unit steps along the x axis at each point along the circle's edge we need to choose between two possible y coordinates

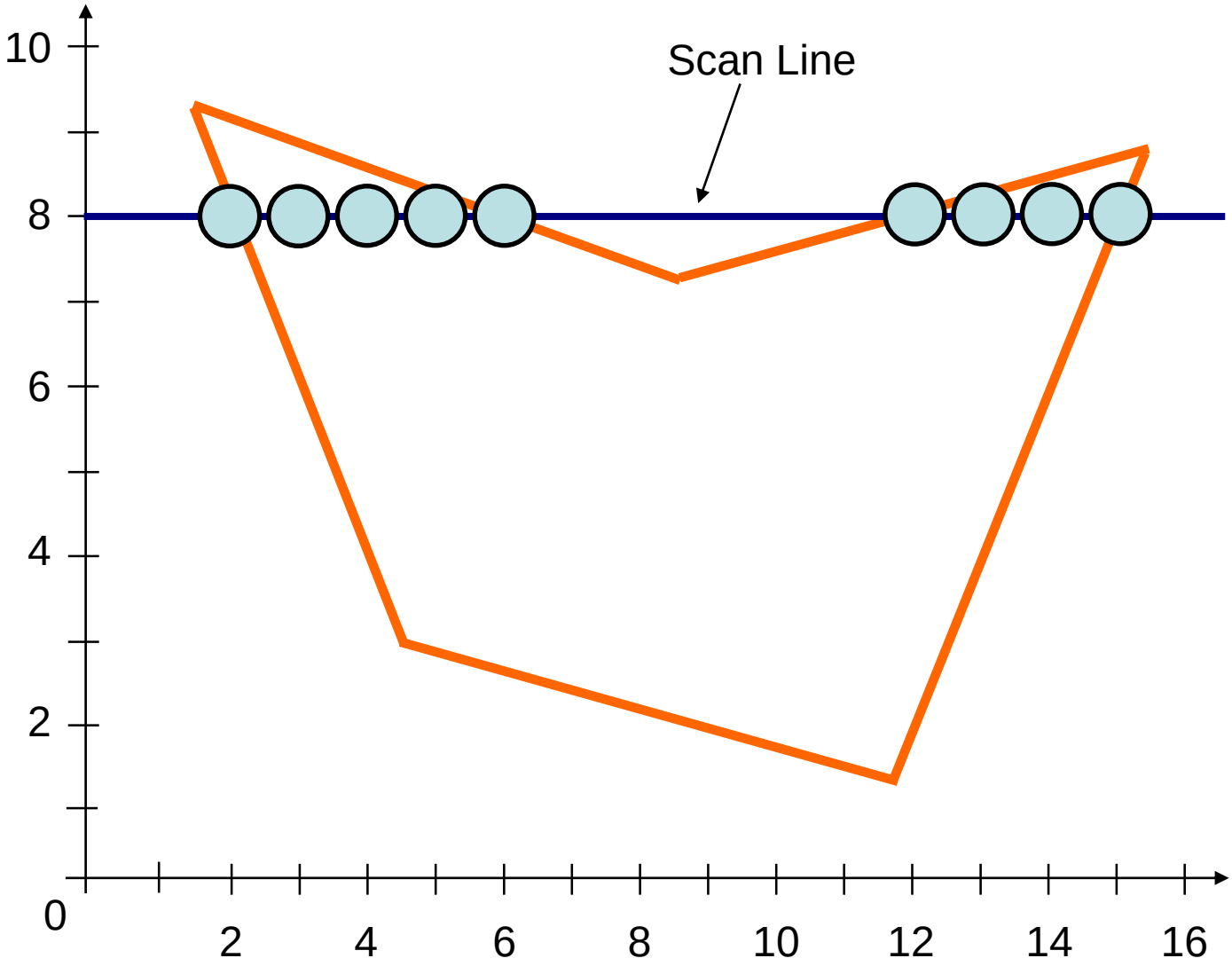
Олон өнцөгтийг дүүргэх

Одоо бид хэрхэн шулуун ба тойрог зурах талаар мэднэ.

Олон өнцөгтийг хэрхэн яаж зурах вэ?

Бид scan-line алгоритм гэж нэрлэгддэг өсөлтийн алгоритмыг ашигладаг

Scan-Line Polygon Fill Algorithm

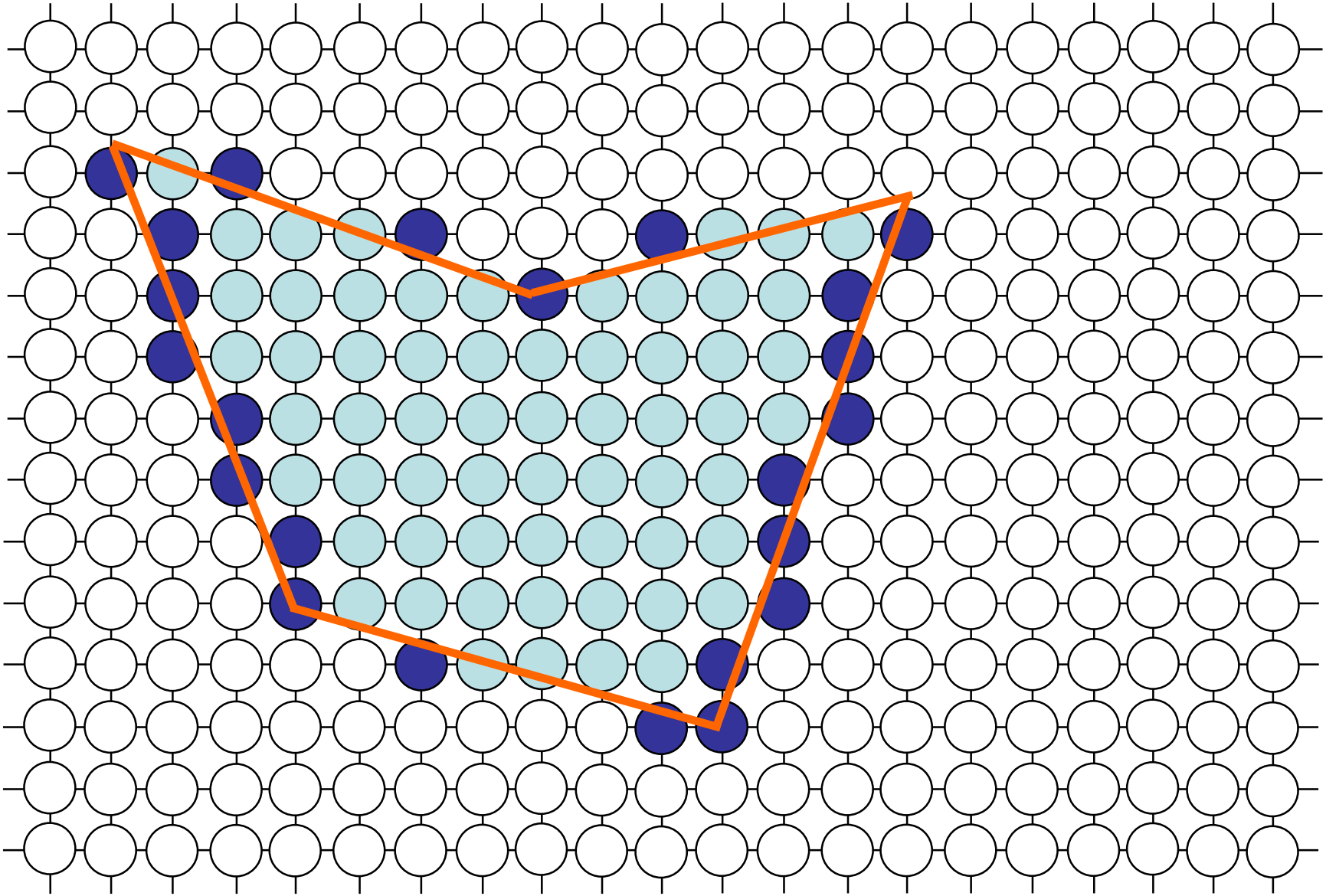


Scan-Line Polygon Fill Algorithm

scan-line үндсэн алгоритм дараах байдалтай:

- Олон өнцөгтийн бүх ирмэгүүдтэй scan line огтлолцлын цэг (intersection) олно
- x координатыг нэмэгдүүлэх замаар огтлолцлын цэг эрэмбэлнэ
- Олон өнцөгтийн дотор талд байрлах огтлолцол хоорондох бүх пикселийг бөглөнө

Scan-Line Polygon Fill Algorithm (cont…)



Шулуун зурах дүгнэлт

Тус лекцээр scan converting lines талаар авч үзлээ.

Бидний анхаарах зүйл бол **FAST**

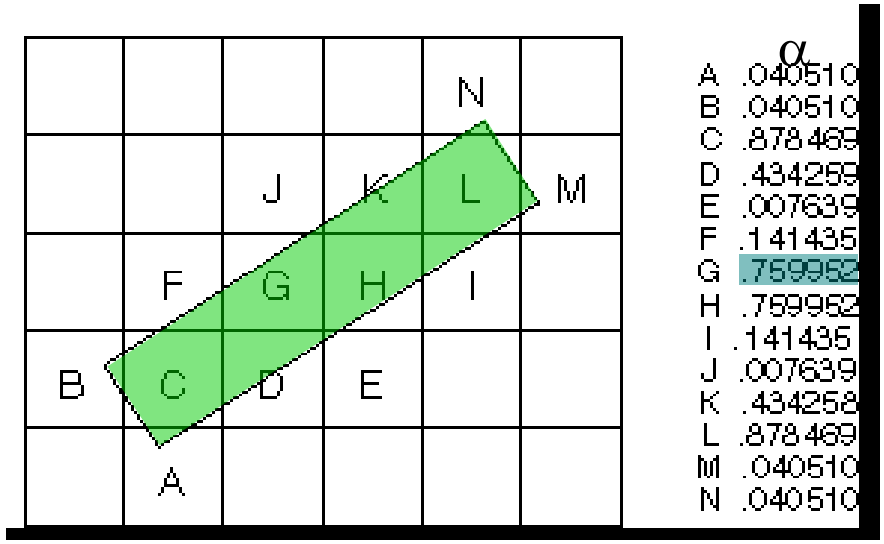
Шулуун зурахад бид DDA эсвэл Bresenham алгоритмыг сонгоно

Тойргийн хувьд mid-point algorithm хэрэгжүүлнэ

Anti-Aliasing

Antialiasing

Conceive that a line is 1 pixel wide which covers certain pixel squares.



Зурах энгийн арга нь C, G, H болон L пикселийг одоогийн өнгөөр будна.

antialiasing арга нь шулуун ба ойролцоох пикселийн квадратуудын давхардсан харьцааг тооцоолно. Тодорхой пикселийн давхцаж буй харьцаа α гэж үзье. Шинэ pixel_color гэж

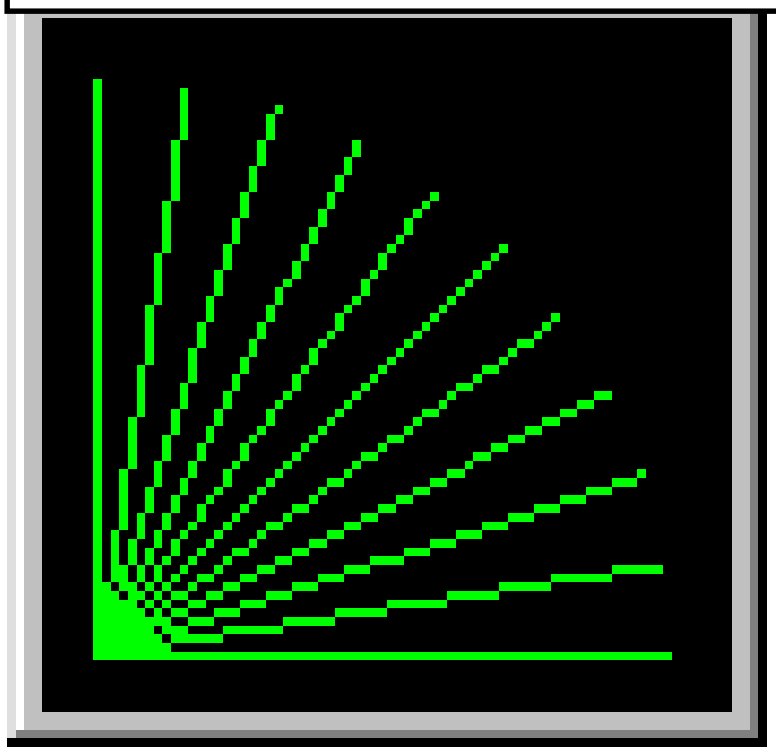
$$\alpha \times current\_color + (1- \alpha) \times existing\_color$$

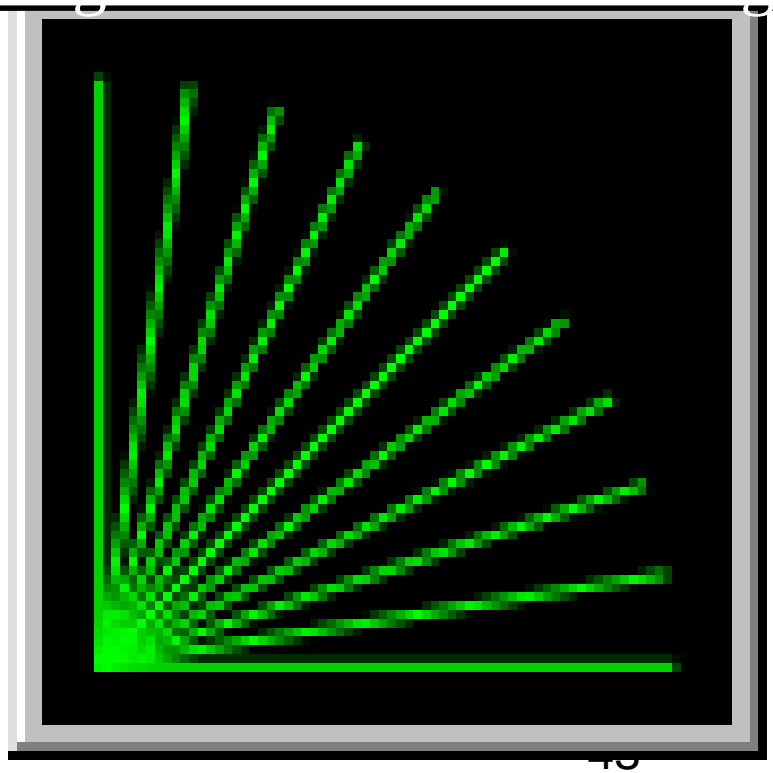
To turn on anitaliasing (draw slower)

enable(LINE_SMOOTH)

To turn off antialiasing (draw faster)

disable(LINE_SMOOTH)





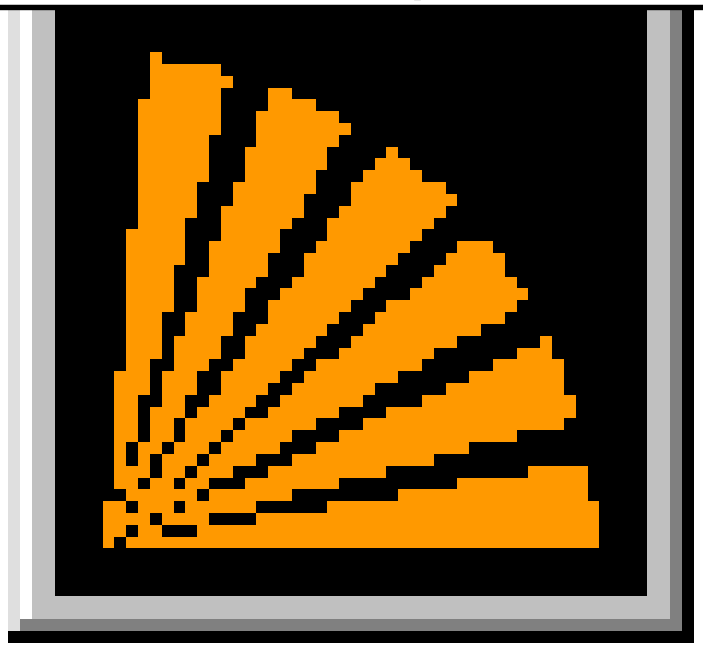
Jaggies нь ирмэгийн дагуу тохиолдож болох ба зураастай адил тэгшлэх боломжтой

To turn on anitaliasing (draw much slower)

enable(POLYGON_SMOOTH)

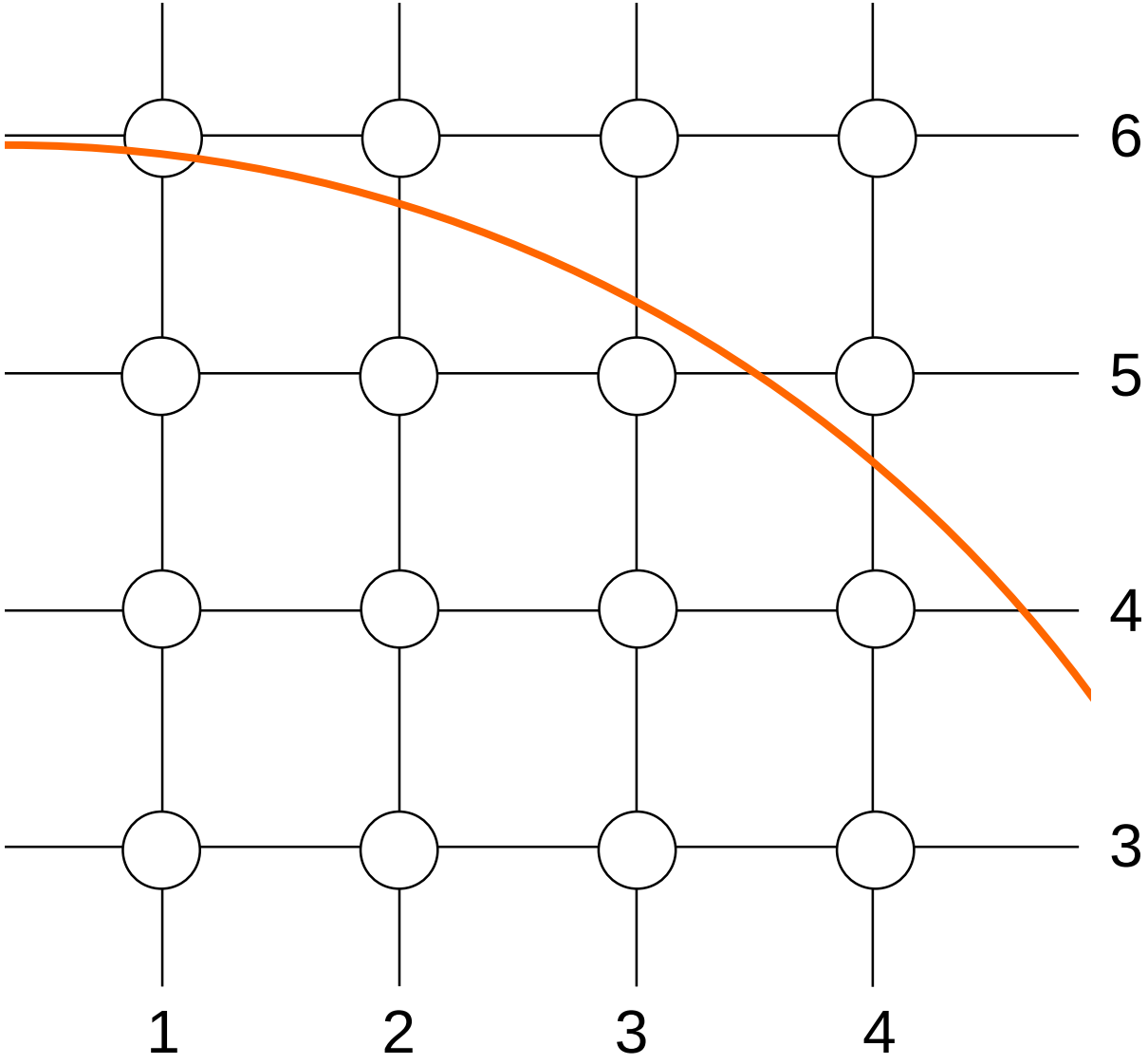
– To turn off antialiasing (draw much faster)

disable(POLYGON_SMOOTH)

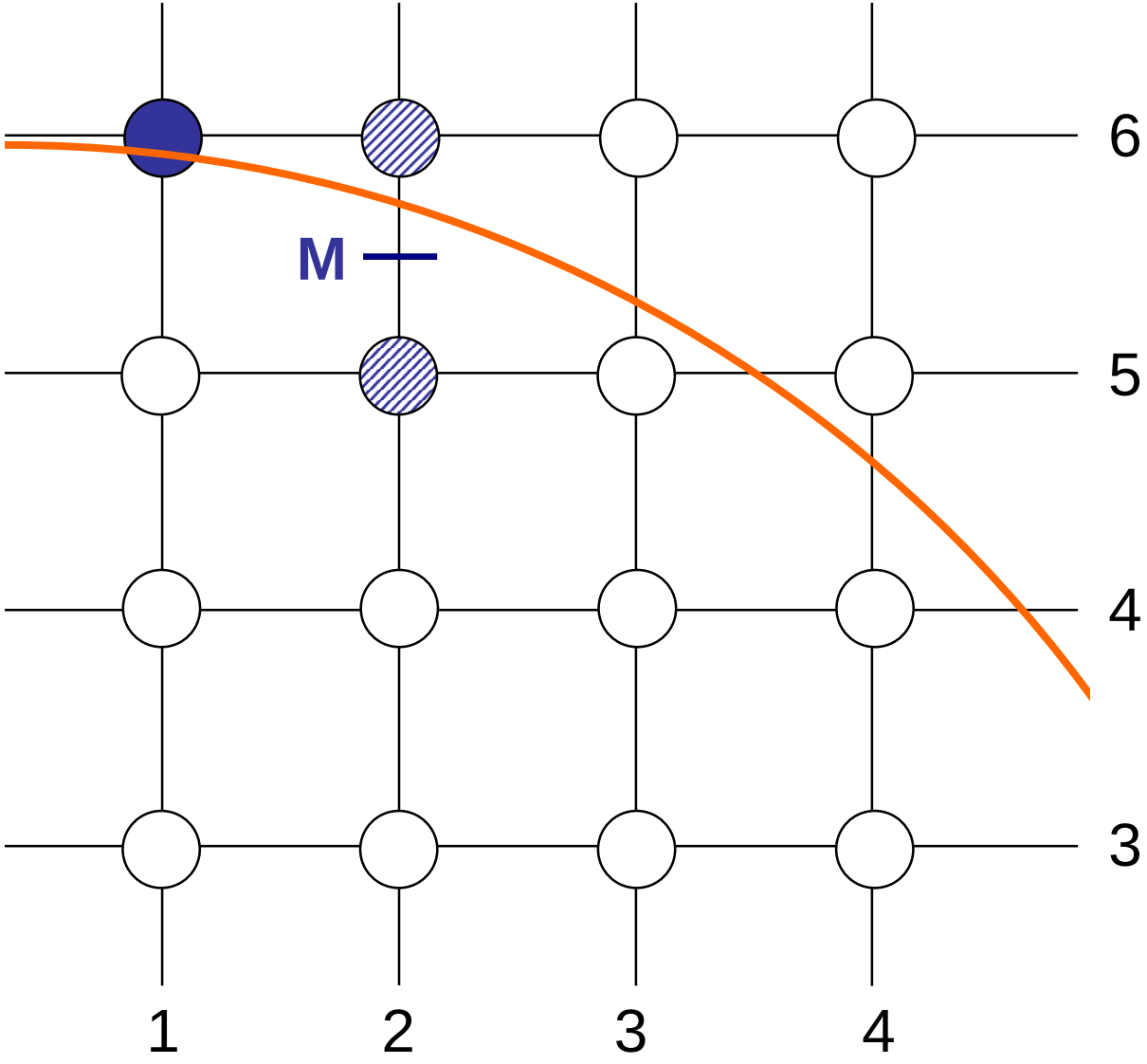


Summary Of Drawing Algorithms

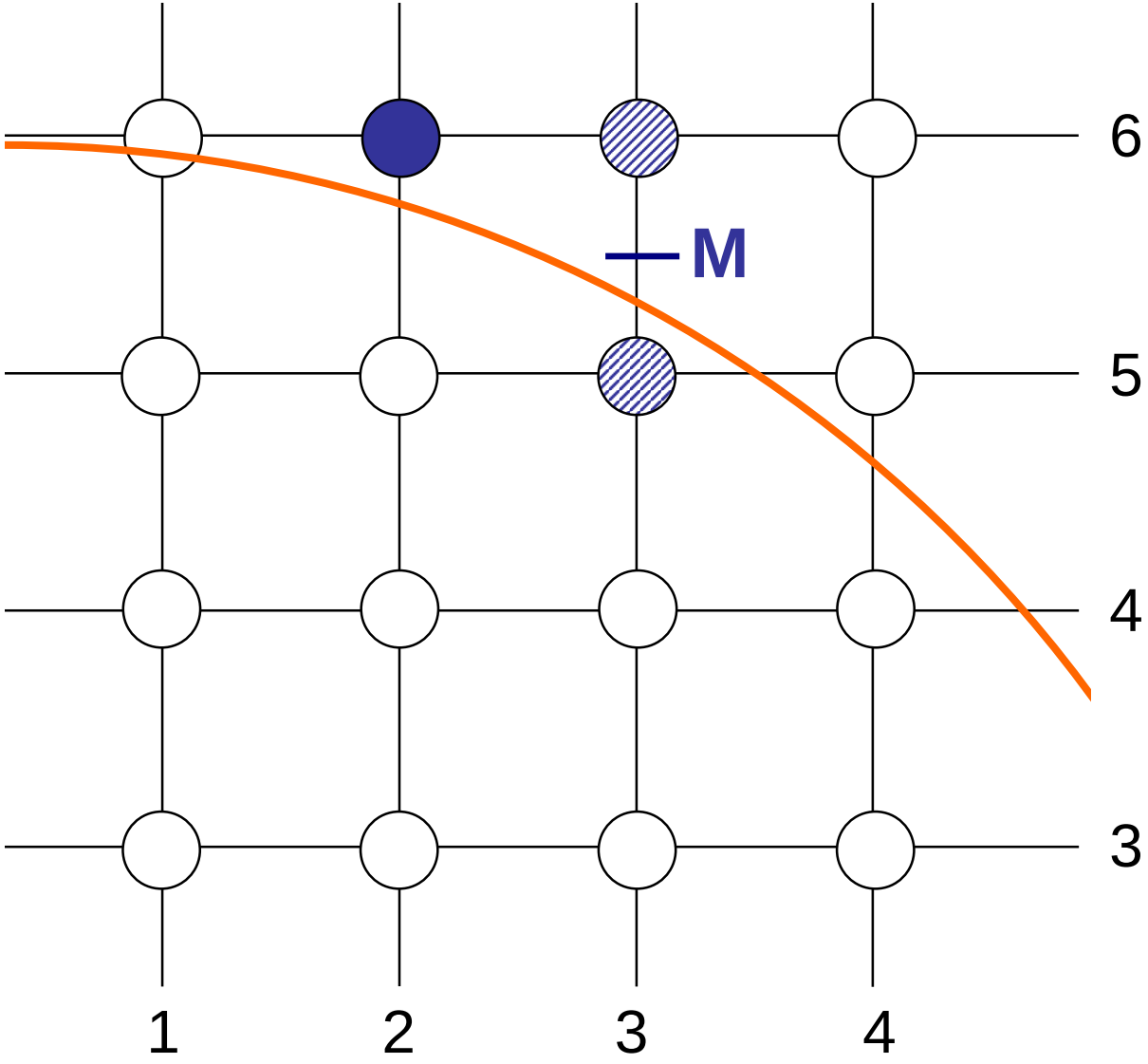
Mid-Point Circle Algorithm (cont…)



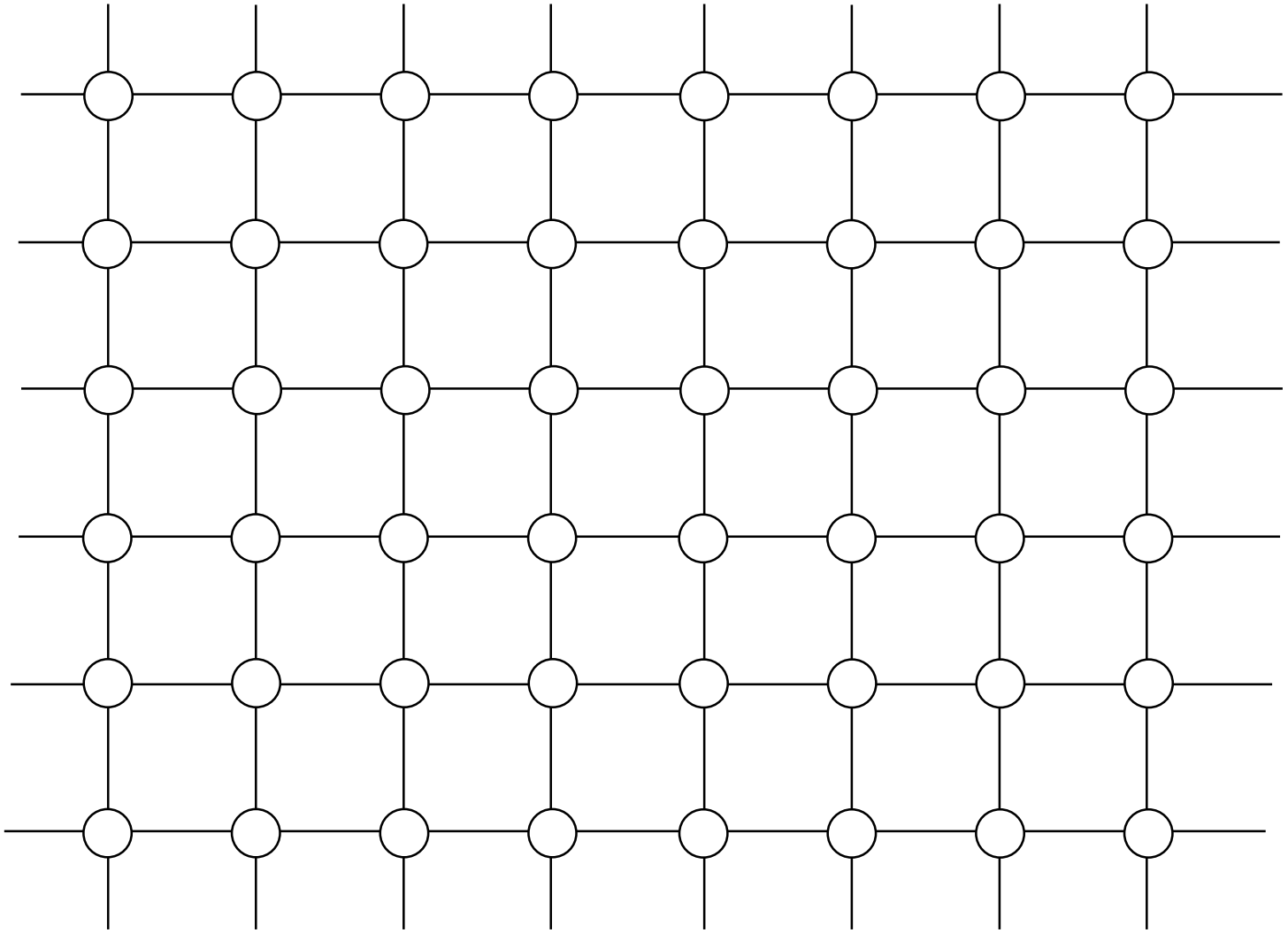
Mid-Point Circle Algorithm (cont…)



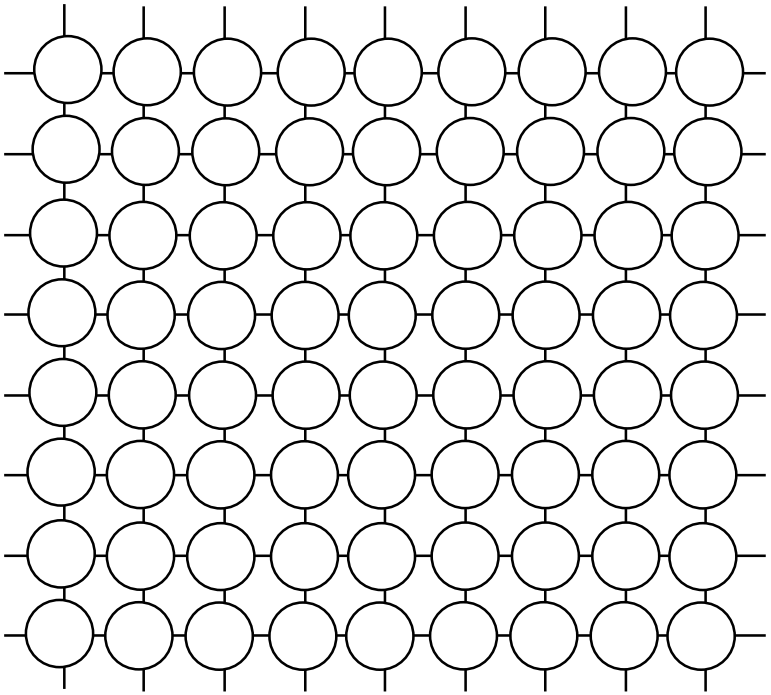
Mid-Point Circle Algorithm (cont…)



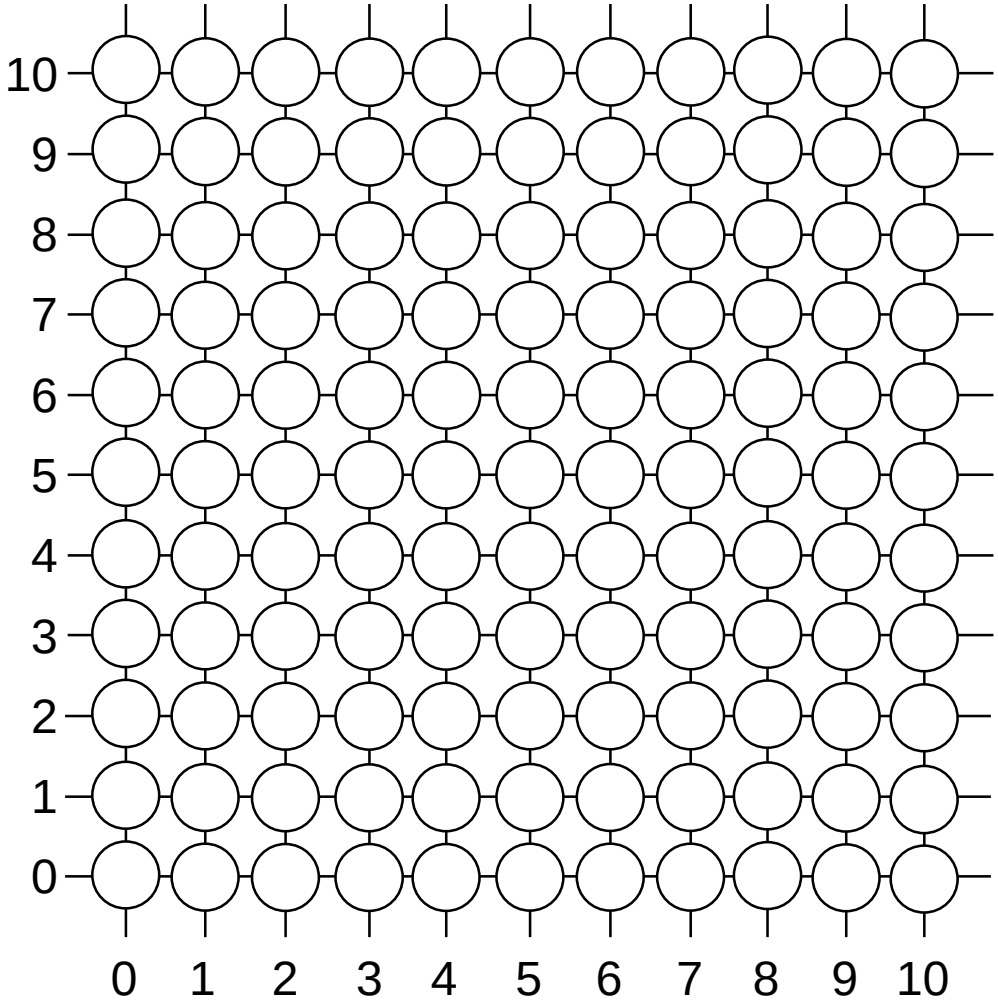
Blank Grid



Blank Grid



Blank Grid



Blank Grid