Exercise Computer graphics

Ultra-fast line drawing

Exercise 4: In the lecture we got to know the mid-point line-drawing algorithm which can generate line pixels with a single comparison and two or three integer additions per pixel, only. Alter the algorithm such that points are plotted more sparsely. Therefore, the algorithm's loop should advance by two pixels in the X-direction rather than one as done in the lecture. The sketch on the right shows the cases which can evolve.

Hints: Start with the initialization of the distance-value at (x0+2,y0+0,5) and (x0+2,y0+1,5) and find out the increments in the inner loop, exactly as done in the normal mid-point algorithm. Note that now you have to take more than one distance into account.

Solution:

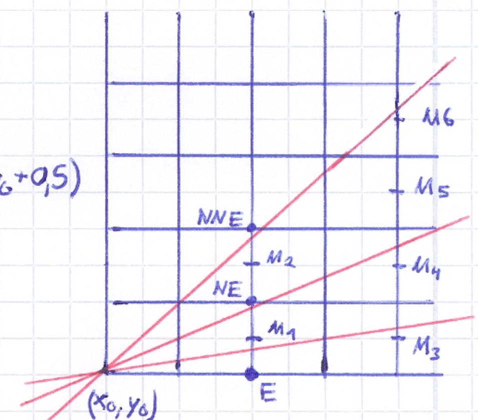
Initialization:

Distance line to M1:

$$d_1 = f(x_0+2, y_0+0{,}5) = \Delta y(x_0+2) - \Delta x(y_0+0{,}5) + \Delta x b$$

$$= \Delta y x_0 + 2\Delta y - \Delta x y_0 - \Delta x\, 0{,}5 + \Delta x b$$

$$= f(x_0, y_0) + 2\Delta y - 0{,}5\Delta x$$



Distance line to M2:

$$d_2 = f(x_0+2, y_0+1{,}5) = \Delta y(x_0+2) - \Delta x(y_0+1{,}5) + \Delta x b$$

$$= \Delta y x_0 + 2\Delta y - \Delta x y_0 - 1{,}5\Delta x + \Delta x b$$

$$= f(x_0, y_0) + 2\Delta y - 1{,}5\Delta x$$

$M_1 = (x_0+2, y_0+0{,}5)$
$M_2 = (x_0+2, y_0+1{,}5)$
$M_3 = (x_0+4, y_0+0{,}5)$
$M_4 = \dots$
$M_5 = \dots$
$M_6 = \dots$

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Exercise Computer graphics

Ultra-fast line drawing

Exercise 4: Solution continued:

- Case 1 : Line above M_1 and M_2 $\Leftrightarrow d_1 > 0;\ d_2 > 0$
- Case 2 : Line between M_1 and M_2 $\Leftrightarrow d_1 > 0;\ d_2 < 0$
- Case 3 : Line below M_1 and M_2 $\Leftrightarrow d_1 < 0;\ d_2 < 0$

Case 1: $x = x + 2;\ y = y + 2$

[In the programm loop we only need to increment d_1 by $2\Delta y - 2\Delta x$]

Distance line to M_5:

$$d_1 = f(x_0+4, y_0+2,5) = f(x_0, y_0) + 4\Delta y - 2,5\Delta x$$
$$= \underbrace{f(x_0+2, y_0+0,5)}_{\text{former value of } d_1} + 2\Delta y - 2\Delta x$$

Distance line to M_6:

$$d_2 = f(x_0+4, y_0+3,5) = \underbrace{f(x_0+2, y_0+1,5)}_{\text{former value of } d_2} + 2\Delta y - 2\Delta x$$

Case 2: $x = x + 2;\ y = y + 1$

Distance line to M_4:

$$d_1 = f(x_0+4, y_0+1,5) = f(x_0+2, y_0+0,5) + 2\Delta y - \Delta x$$

Distance line to M_5:

$$d_2 = f(x_0+4, y_0+2,5) = f(x_0+2, y_0+1,5) + 2\Delta y - \Delta x$$

Case 3: $x = x + 2;$

Distance line to M_3:

$$d_1 = f(x_0+4, y_0+0,5) = f(x_0+2, y_0+0,5) + 2\Delta y$$

Distance line to M_4:

$$d_2 = f(x_0+4, y_0+1,5) = f(x_0+2, y_0+1,5) + 2\Delta y$$

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Exercise Computer graphics

Ultra-fast line drawing

Exercise 5: The mid-point algorithm uses integer additions only. Usually, 16 bits are sufficient for representing either x- or y-components. However, modern processors are operating at 32 or even 64 bits at a time. Assuming that an integer has at least 32 bits, we exploit the lower 16 bits only because resolutions of more than 65535 x 65535 pixels are unlikely.

Change the midpoint implementation such that the most significant 16 bits are used as well. Which improvements can you achieve?

Solution: Here, the y-component and the distance is processed by a single variable “value”.

```
value = y0 << 16 | dist;

SetPixel(screen, x0, y0);

for(int x = x0; x <= x1; x++) {
  if((value & 0xFFFF) < 0) value += incE;
  else value += incNE | (1 << 16);

  SetPixel(screen, x, value >> 16);
} // while
```

An alternative solution would be to draw two lines within a single loop until the shorter line is finished. The rest of the longer line should then be drawn in a subsequent loop (see tared-code for details).

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Exercise Computer graphics

Ultra-fast line drawing

Solution: An alternative solution would be to draw two lines within a single loop until the shorter line is finished. The rest of the longer line should then be drawn in a subsequent loop (see tared-code for details).

```
x = xA | ((long)xB << 16);
y = yA | ((long)yB << 16);

for(int i = 0; i < min(lengthA,lengthB); i++) {

  if(distA < 0) distA += incEA;
  else {
    distA += incNEA;
    y++;
  } // else

  if(distB < 0) distB += incEB;
  else {
    distB += incNEB;
    y += 1L << 16;
  } // else

  x += (1 | (1L << 16));

  SetPixel(screen, x & 0xFFFF, y & 0xFFFF);
  SetPixel(screen, x >> 16, y >> 16);
} // for
```

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim