Einfache Kurven auf Rastergrafiken

David Kastrup

12. Juli 2013

Zusammenfassung

Es sollen hier einfache Methoden vorgestellt werden, um auf einer Rastereinheit verschiedene Kurven darzustellen. Vorgestellt werden Zeichenalgorithmen fr Linien, Kreise und Hyperbeln. Die hier hergeleiteten Gleichungen sind auch unter dem Namen `DDA`s bekannt.

1 Einfhrung

Bei den hier vorgestellten Algorithmen werden zunchst nur Kurvenstcke betrachtet, die die folgenden Eigenschaften besitzen:

1. Sie lassen sich als Funktion $y = f(x)$ darstellen.
2. y ist im betrachteten Bereich monoton, das heit, entweder durchgehend steigend oder durchgehend fallend.
3. Wenn x sich um 1 ndert, so ndert sich y betragsmig hchstens um 1 ($\left|\frac{\partial y}{\partial x}\right| \leq 1$).

2 Die gerade Linie

Wir betrachten hier zunchst nur die gerade Linie im ersten Oktanten, die durch den Punkt $\binom{0}{0}$ geht. Alle anderen Linien lassen sich durch Vertauschen von x und y sowie Vorzeichenwechsel erzeugen. Im ersten Oktanten gilt $x \geq y \geq 0$. Zum Zeichnen einer Linie gengt es also, x durchlaufen zu lassen und fr y die dazugehrigen Werte zu berechnen und zu runden.

Die Gleichung einer Geraden durch $\binom{\Delta x}{\Delta y}$ lautet:

$$y = \frac{\Delta y}{\Delta x} x \tag{1}$$

Nun stellen wir y als Summe eines ganzzahligen Wertes e und eines gebrochenen Wertes f dar, fr den gilt: $-0.5 \leq f < 0.5$. Somit stellt dann e den gewnschten, auf die nchste ganze Zahl gerundeten y-Wert dar. Jetzt formen wir (??) um:

$$\begin{aligned} e + f &= x\frac{\Delta y}{\Delta x} \\ e\Delta x + f\Delta x &= x\Delta y \\ f\Delta x - \left\lceil \frac{\Delta x}{2} \right\rceil &= x\Delta y - e\Delta x - \left\lceil \frac{\Delta x}{2} \right\rceil \end{aligned} \tag{2}$$

Den linken Ausdruck in (??) bezeichnen wir jetzt mit d. Fr positive gerade Werte von Δx ist offensichtlich $d < 0$ eine zu $f < 0.5$ equivalente Bedingung.

Fr ungerade Werte von Δx ist $f < 0.5$ equivalent zu $d + 0.5 < 0$. Da d stets eine ganze Zahl ist, ist dies wieder zu $d < 0$ equivalent.

Wird nun $f \geq 0.5$, wie sich durch den Vergleich $d \overset{?}{<} 0$ feststellen lt, so mu man korrigieren, indem man f um 1 erniedrigt und e um 1 erhht. d mu dann auch entsprechend angepat werden.

Mit den angegebenen Formeln ergibt sich jetzt bei Bercksichtigung der Einflsse von x und e auf d der in Tabelle **??** angegebene Algorithmus. Eine optimierte C-function, die die Oktantenaufteilung bercksichtigt, ist in Tabelle **??** zu finden. Einige hiermit gezeichnete Linien sind in Abbildung **??** zu sehen.

3 Der Kreis

Wir betrachten hier nur den Achtelkreis im zweiten Oktanten ($y \geq x \geq 0$). Hier gelten die oben angegebenen Beziehungen. Alle anderen Achtelkreise lassen sich durch elementare Spiegelungen erzeugen.

Die Gleichung eines Kreises ist hier

$$y = \pm\sqrt{r^2 - x^2} \tag{3}$$

Tabelle 1: Linienzugalgorithmus

1. Setze $x \leftarrow 0$, $y \leftarrow 0$, $d \leftarrow -\left\lceil \frac{\Delta x}{2} \right\rceil$
2. Wiederhole bis $x = \Delta x$
   (a) Zeichne Punkt an $\binom{x}{y}$
   (b) Setze $x \leftarrow x + 1$, $d \leftarrow d + \Delta y$
   (c) Falls $d \geq 0$
      i. Setze $d \leftarrow d - \Delta x$
      ii. Setze $y \leftarrow y + 1$

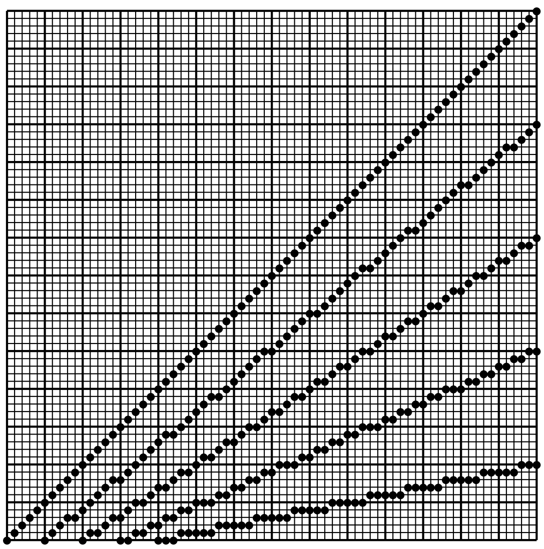

Abbildung 1: Einige Linien

Tabelle 2: Linienziehen in C

```
extern int x,y;
/* (x,y) ist Koordinate des nicht
 * gezeichneten Startpunktes, zeigt
 * nachher auf gezeichneten Endpunkt
 */
#define doline(dx,dy,advx,advy) { \
  d = -(((i = dx) + 1) >> 1); \
  while (i--) { \
    advx; \
    if ((d += dy) >= 0) { \
      d -= dx; advy; \
    } \
    dot(x,y); \
  } \
  return; \
} /* Grundalgorithmus 1. Oktant */
/* dx ist Distanz in unabh. Richtung, *
 * dy in abh. Richtung, advx geht     *
 * in unabh. Richtung, advy in abh.   */

#define docond(cond,advx,advy) { \
  if (cond) doline(dx,dy,advx,advy) \
  doline(dy,dx,advy,advx) \
} /* Grundalgorithmus 1./2. Oktant */
/* cond ist true falls |dx| > |dy| */

void
linedraw(int dx, int dy)
/* Von (x,y) nach (x+dx, y+dx). */
{
  int i;

  if (dx >= 0) {
    if (dy >= 0)
      docond(dx > dy, ++x, ++y)
    docond(dx > (dy = -dy), ++x, --y)
  }
  if (dy >= 0)
    docond((dx = -dx) > dy,--x,++y)
  docond((dx = -dx) > (dy = -dy),
           --x, --y )
}
```

Der Wert y lt sich darstellen als Summe einer ganzen Zahl e und einem Wert f mit $-0.5 \leq f < 0.5$. Der Wertebereich von f ist so gewhlt worden, damit e einen auf ganze Zahlen gerundeten Wert fr y darstellt.

Nun gilt:

$$
\begin{aligned}
e + f &= \sqrt{r^2 - x^2} \\
e^2 + 2ef + f^2 &= r^2 - x^2 \qquad (4)
\end{aligned}
$$

Die Gleichung (??) hat fr $x+1$ folgende Form:

$$
e'^2 + 2e'f' + f'^2 = r^2 - x^2 - 2x - 1 \qquad (5)
$$

Zieht man die Gleichung (??) von (??) ab, so ergibt sich nach Umsortieren:

$$
\begin{aligned}
&e' = e : \\
&2e'f' + f'^2 = 2ef + f^2 - 2x - 1 \\
&e' = e - 1 : \\
&2e'f' + f'^2 = 2ef + f^2 + 2e - 2x - 2
\end{aligned}
$$

Jetzt wird $2ef + f^2$ mit m getauft. Also:

$$
\begin{aligned}
&e' = e : \\
&m' = m - 2x - 1 \\
&e' = e - 1 : \\
&m' = m + 2e - 1 - 2x - 1
\end{aligned}
$$

Wie gro ist jetzt m? Fr $x = 0$ ist es sicher 0. Solange e konstant bleibt, schrumpft f stetig. Fllt f unter -0.5, so fllt m (unter Vernachlssigung von f^2) unter $-e$. Dies wird jetzt als Kriterium fr einen Unterlauf von f verwendet. Tritt dieser auf, so mu f um 1 erhht und e um 1 erniedrigt werden.

Um die Abfragebedingung zu vereinfachen, setzt man jetzt $q = m + e$. Der resultierende Algorithmus ist in Tabelle ??, ein optimiertes C-Programm ist in Tabelle ?? zu finden.

4 Einfache Hyperbeln

Als letztes Beispiel betrachten wir hier Hyperbeln, die der Formel $y = r^2/x$ gengen, und zwar im Bereich $x \geq r$.

Mit dem Ansatz $y = e + f$ ergibt sich:

$$
\begin{aligned}
e + f &= r^2/x \\
ex + fx &= r^2 \\
fx &= r^2 - ex \qquad (6)
\end{aligned}
$$

Fr $x' = x + 1$ hat nun (??) die Form

Tabelle 3: Kreiszeichenalgorithmus

1. Setze $x \leftarrow 0$, $y \leftarrow r$, $q \leftarrow r$
2. Wiederhole bis $x > y$:
   (a) Setze einen Punkt an $\binom{x}{y}$.
   (b) Setze $q \leftarrow q - 2x - 1$
   (c) Falls $q < 0$
      i. Setze $q \leftarrow q + 2y - 2$
      ii. Setze $y \leftarrow y - 1$
   (d) Setze $x \leftarrow x + 1$

Tabelle 4: Kreiszeichenprogramm

```
void
fourfold(int x0, int y0, int x, int y)
/* Zeichne in Oktant 1,3,5,7 */
/* Wird benutzt, um Anfangs- und End- *
 * Punkte nicht zweimal zu zeichnen   */
{
  dot(x0+x,y0+y);
  dot(x0-y,y0+x);
  dot(x0-x,y0-y);
  dot(x0+y,y0-x);
}

void
eightfold(int x0, int y0, int x, int y)
/* Zeichne in allen Quadranten */
{
  fourfold(x0,y0,x,y);  /* 1357 */
  fourfold(x0,y0,x,-y); /* 8642 */
}

void
circle(int x0, int y0, int r)
{
  fourfold(x0,y0,0,r);
  /* Die ersten vier Punkte */
  for (x=0, y=q=r;; ) {
    if ((q -= 2* x++ + 1) < 0)
      q += 2* --y;
    if (x >= y)
      break;
    eightfold(x0,y0,x,y);
  }
  if (x==y)
    fourfold(x0,y0,x,y);
  /* Eventuell die letzten vier */
}
```

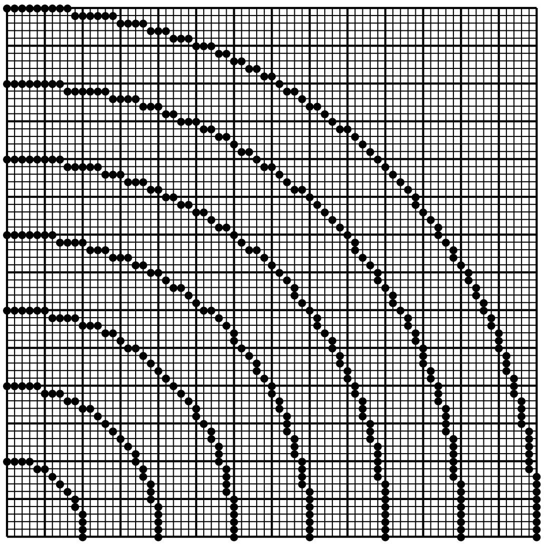

Abbildung 2: Viertelkreise

$$
\begin{aligned}
e' = e: & \\
f'x' &= r^2 - ex - e \\
e' = e - 1: & \\
f'x' &= r^2 - ex - e + x + 1
\end{aligned}
$$

Setzt man jetzt $d = (2f + 1)x$, so ist $f < -0.5$ mit $d < 0$ equivalent. Erreicht man diesen Fall unter Verwendung der Annahme $e' = e$, dann mu in bekannter Weise f um 1 erhht und e um 1 vermindert werden.

Tabelle 5: Hyperbelalgorithmus

1. Setze $d \leftarrow r$, $x \leftarrow r$, $y \leftarrow r$
2. Wiederhole bis zufrieden
   (a) Setze Punkt an $\binom{x}{y}$
   (b) Setze $x \leftarrow x + 1$
   (c) Setze $d \leftarrow d - 2y + 1$
   (d) Falls $d < 0$
      i. Setze $d \leftarrow d + 2x$
      ii. Setze $y \leftarrow y - 1$

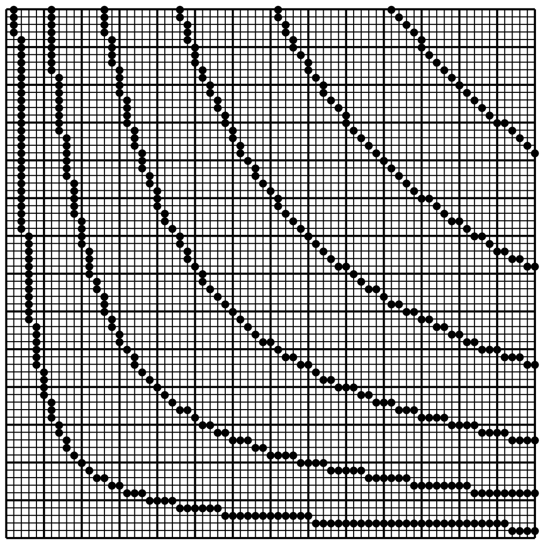

Abbildung 3: Hyperbeln

Fr d' ergeben sich dann die folgenden Werte:

$$
\begin{aligned}
e' = e: & \\
d' &= d - 2e + 1 \\
e' = e - 1: & \\
d' &= d - 2e + 2x + 2 + 1
\end{aligned}
$$

Daraus ergibt sich der in Tabelle **??** angegebene Hyperbelalgorithmus fr den ersten Oktanten.

Tabelle 6: Hyperbeln in C

```
void
four(int x0, int y0, int x, int y)
/* Hyperbeln sind nur in 4 Oktanten */
{
  dot(x0+x,y0+y);
  dot(x0+y,y0+x);
  dot(x0-x,y0-y);
  dot(x0-y,y0-x);
}

void
hyperb(int x0, int y0, int r, int dx)
{
  int d, x, y;

  for (x = y = d = r; dx--;) {
    four(x0,y0,x,y);
    ++x;
    if ((d -= 2*y + 1) < 0) {
      d += 2*x;
      --y;
    }
  }
}
```