Line Drawing, Leap Years, and Euclid

MITCHELL A. HARRIS

Technical University of Dresden

AND

EDWARD M. REINGOLD

Illinois Institute of Technology

Abstract. Bresenham's algorithm minimizes error in drawing lines on integer grid points; leap year calculations, surprisingly, are a generalization. We compare the two calculations, explicate the pattern, and discuss the connection of the leap year/line pattern with integer division and Euclid's algorithm for computing the greatest common divisor.

Categories and Subject Descriptors: F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Number-theoretic computations* (*e.g., factoring, primality testing*); I.3.3 [**Computer Graphics**]: Picture/Image Generation; J.2 [**Physical Sciences and Engineering**]: *Astronomy*; *Mathematics and statistics*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Bresenham's algorithm, calendar algorithms, continued fractions, Euclid's algorithm, greatest common divisor, leap years, line drawing, scan-line conversion

1. INTRODUCTION

Bresenham's algorithm [Bresenham 1965] is the classic technique for plotting lines on bitmaps: it approximates linear segments defined by rational coefficients using only integral points. The algorithm calculates a finite set of points on the integer lattice with minimum total vertical distance from the original line segment. The essential design features of the algorithm are minimization of error (it converts a line segment from a continuous domain to one that is discrete with as little error as possible), speed (it uses only integer arithmetic), and parsimony (as few pixels as possible are used to ensure adjacency).

In a very different domain, leap year calculations [Reingold and Dershowitz 2001] involve a similar mapping from the continuous to the discrete. By astronomical observation, one can calculate the ratio between the duration of the revolution of the earth around the sun (the year), and the duration of the rotation of the earth (the day); there are approximately

Authors' addresses: M. A. Harris, Department of Computer Science, Technical University of Dresden, D-01062 Dresden, Germany; email: harris@tcs.inf.tu-dresden.de; E. M. Reingold, Department of Computer Science, Illinois Institute of Technology, Stuart Building, 10 West 31st Street, Suite 236, Chicago, IL 60616-2987; email: reingold@iit.edu.

Permission to make digital/hard copy of part or all of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication, and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
©2004 ACM 0360-0300/04/0300-0068 $5.00

365.25 days per year. But calendars, by design, require an integral number of days per year; to account for the extra quarter of a day, we occasionally add an extra day to the year. Arithmetical calendars such as the Julian and the Jewish, and the common arithmetical approximation to the Islamic calendar, have special formulæ for specifying exactly which years get extra days; these calendars add extra days as evenly as possible over the course of years—we will see what "evenly" means in the discussion of calendars in Section 2.

How are the line segment and the leap year calculations related? It turns out that Bresenham's algorithm computes a special case of leap year calculations. They are both computing approximations to a line with rational slope using integer division. Both the repeating pattern of dots in a rasterized line segment and the repeating pattern of leap years can be computed using integer division—and they are intimately related to Euclid's algorithm for calculating greatest common divisors and continued fractions. Not surprisingly, others have noticed similarities among these three areas. The connection between astronomical cycles and greatest common divisors is folkloric [Rockett and Szüsz 1992], going even back to the Greeks. More recently, Brons [1974], Castle and Pitteway [1987], Pitteway [1985], and Troesch [1998] all discussed the connection between line drawing and Euclid's algorithm. Here we give explicit correspondences among all three. First, we discuss the details of each domain, and then compare them.

2. LINE DRAWING

In this section we derive Bresenham's line drawing algorithm, one of the fundamental algorithms of computer graphics. In its final form, as usually presented, the algorithm is obscure; deriving it in a step-by-step way is both interesting and illuminating. Furthermore, it is much easier to deal with the simpler initial form than with the equivalent final form.

A line drawing algorithm has as its input the integer starting point (x_S, y_S) and the integer ending point (x_E, y_E) of a line segment. The basic algorithm calculates the slope of the line,

$$\frac{\Delta y}{\Delta x} = \frac{y_E - y_S}{x_E - x_S}.$$

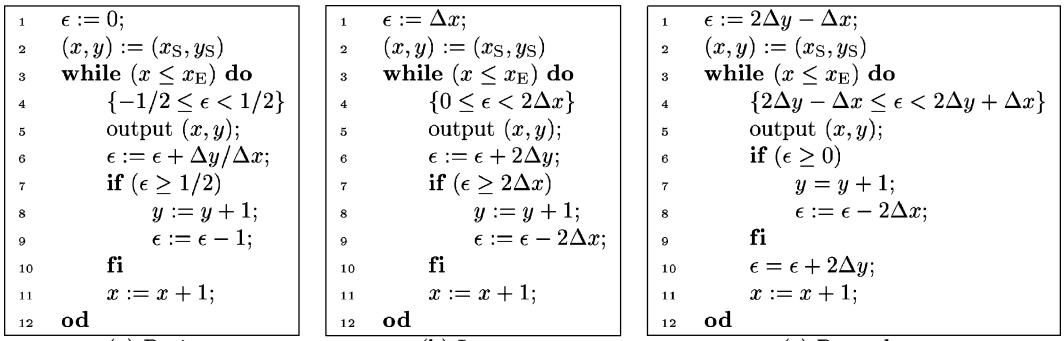
For simplicity we assume that $0 < \Delta y \leq \Delta x \neq 0$: if either $\Delta x = 0$ or $\Delta y = 0$, the line drawing is trivial; if $\Delta y > \Delta x$, we merely interchange the x and y axes. To draw the line segment, we want to darken the $x_E - x_S + 1$ pixels specified by

$$\begin{aligned} y &= y_S + \text{round}\left(\frac{\Delta y}{\Delta x}(x - x_S)\right), \\ x &= x_S, x_S + 1, \ldots, x_E. \end{aligned} \tag{1}$$

Traditionally, this sequence of pixels is encoded as a sequence of the plotter controls "east" and "northeast." We prefer to separate the changes of x and y into just "east" and "north" movements.

We start with $(x, y) = (x_S, y_S)$, and calculate the sequence of pixels incrementally, repeatedly adding 1 to x, and occasionally adding 1 to y. We make the decision to increment y by keeping track of the error ϵ, the vertical distance of the pixel to the true line segment. If $\epsilon \geq 1/2$, the pixel above more closely approximates the true value of y, so we increment y by 1 and decrement ϵ by 1. Otherwise, we add the slope $\Delta y/\Delta x$ (which is between 0 and 1 by assumption) to ϵ and do not change y. The initial error is $\epsilon = 0$ and this algorithm minimizes the total vertical error of the pixels to the real line, since inductively the error ϵ is always in the range $[-1/2, 1/2]$, the best possible on the integer lattice. This algorithm is shown in Algorithm 1(a); it computes the points given by Equation (1).

Algorithm 1(a) can easily be restricted to integer operations—calculate the (rational) slope $\Delta x/\Delta y$ in lowest common terms, so that Δx and Δy are relatively prime, and multiply all assignments and tests involving ϵ by $2\Delta x$; for convenience we also shift the range of ϵ from $[-\Delta x, \Delta x]$ to $[0, 2\Delta x]$; the result is Algorithm 1(b).



```
1   ε := 0;
2   (x, y) := (x_S, y_S)
3   while (x ≤ x_E) do
4       {−1/2 ≤ ε < 1/2}
5       output (x, y);
6       ε := ε + Δy/Δx;
7       if (ε ≥ 1/2)
8           y := y + 1;
9           ε := ε − 1;
10      fi
11      x := x + 1;
12  od
```

(a) Basic

```
1   ε := Δx;
2   (x, y) := (x_S, y_S)
3   while (x ≤ x_E) do
4       {0 ≤ ε < 2Δx}
5       output (x, y);
6       ε := ε + 2Δy;
7       if (ε ≥ 2Δx)
8           y := y + 1;
9           ε := ε − 2Δx;
10      fi
11      x := x + 1;
12  od
```

(b) Integer

```
1   ε := 2Δy − Δx;
2   (x, y) := (x_S, y_S)
3   while (x ≤ x_E) do
4       {2Δy − Δx ≤ ε < 2Δy + Δx}
5       output (x, y);
6       if (ε ≥ 0)
7           y = y + 1;
8           ε := ε − 2Δx;
9       fi
10      ε = ε + 2Δy;
11      x := x + 1;
12  od
```

(c) Bresenham

Algorigthm 1. Transformations leading to Bresenham's algorithm. Version (a) uses floating point operations, eliminated in version (b) by multiplying through by $2\Delta x$ and shifting the resulting range of ϵ from $[-\Delta x, \Delta x)$ to $[0, 2\Delta x)$; version (c) results from then moving the update of the error to after the comparison, shifting the initial value of ϵ, and using a zero test in the **if**.

This algorithm can be converted, in turn, to the traditional Bresenham algorithm of Algorithm 1(c) by moving the update of the error after the comparison, shifting the initial value of ϵ, and using a "zero test" (this simpler test is the point of the transformation). Each of these transformations preserves correctness.

Examining Algorithm 1(c), we see that y is incremented by 1 only when ϵ is decremented by $2\Delta x$, making y the quotient on division by $2\Delta x$, with remainder ϵ. Hence, the equation plotted by the algorithm corresponds to Equation (1):

$$\begin{aligned} y_n &= y_S + \left\lfloor \frac{2n\Delta y + \epsilon_0}{2\Delta x} \right\rfloor \\ &= y_S + \left\lfloor \frac{2(x_n - x_S)\Delta y + \Delta x}{2\Delta x} \right\rfloor, \quad (2) \end{aligned}$$

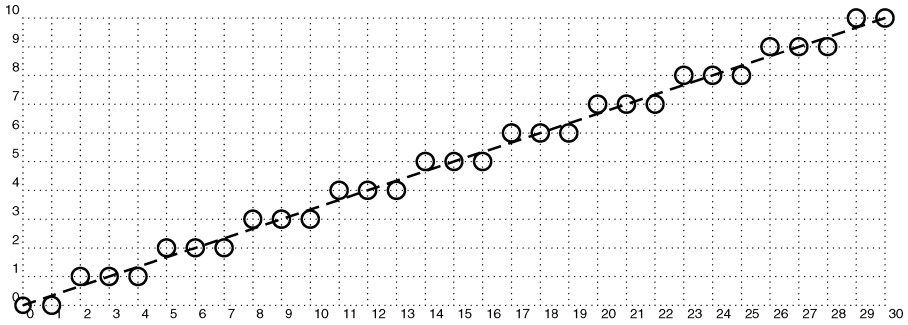
Since at every step an increment is made to x, and possibly to y, the resulting line drawing is "connected" in that successive points are adjacent horizontally or diagonally. As an example, three lines with different right endpoints are shown in Figure 1. Note the simple pattern for the line from $(0, 0)$ to $(30, 10)$ in Figure 1(a) and the more complicated patterns for lines from $(0, 0)$ to $(30, 11)$ and $(30, 12)$ in Figures 1(b) and 1(c), respectively.

We call a set of points generated by Bresenham's algorithm a *Bresenham line*. Because of the manner in which x and y are incremented, every x coordinate and every y coordinate between the endpoints occurs in a Bresenham line. All such lines are made of horizontal segments of length one or more; the lengths of the segments in a particular Bresenham line have no more than two distinct values, excluding the segments touching the endpoints. If $\Delta y = 1$, then the line has a uniform appearance, with a single length for the segments. For $\Delta y > 1$, the pattern of alternating long and short sequences appears to have some pattern but it is difficult to determine by inspection. Changing the initial value of ϵ does not change the basic pattern, but shifts it.

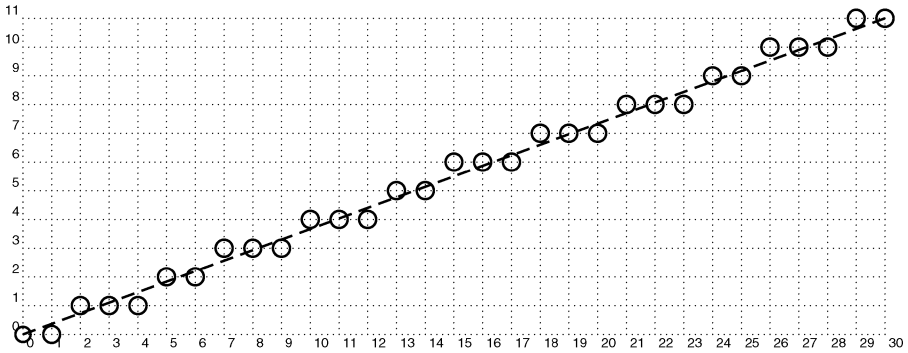
Bresenham's original paper [1965] essentially gives Algorithm 1(c) directly from an analysis of error. Sproull [1982] derived Bresenham's algorithm as we do, starting with the straightforward naïve algorithm, using rationals and transformed it step by step into Bresenham's by scaling and shifting. Berstel [1990] described the Bresenham patterns using formal language theory; these patterns are an instance of *Sturmian words*—see Allouche and Shallit [2003], Section 9.2.

3. LEAP YEARS

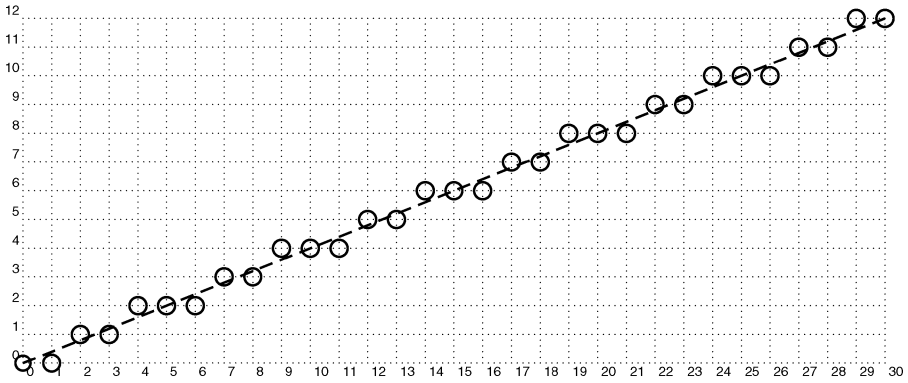
Humans observed millennia ago that there are approximately 365.25 days



(a) $(0, 0)$ to $(30, 10)$



(b) $(0, 0)$ to $(30, 11)$



(c) $(0, 0)$ to $(30, 12)$

Fig. 1. Three examples of lines and their approximation by points using Bresenham's algorithm; the circled lattice points in a plot constitute the "Bresenham line" approximating the dashed line shown.

between successive winter solstices; hence there are nearly 1461 days for every 4 years. Since $1461 - 365 \times 4 = 1$, it takes 4 years for the extra quarter of a day to add up to a full day, so an extra day must be added every four years. This corresponds to the (old style) Julian calendar leap year rule: year y is a leap year if and only if $y \equiv 0 \pmod 4$, and to the Coptic/Ethiopic rule that year y is a leap year if and only if $y \equiv 3 \pmod 4$. The given ratio of 1461 days every 4 years gives an average year length of 365.25 days per year, achieved with an integral number of days per year.

But an average of 365.25 days per year is not fully accurate. The true average (which is changing very slowly over time) is closer to 365.242 days per year and that small difference accumulates into an error of about one day every 125 years. By the sixteenth century the error had caused the spring equinox to shift from its traditional (Roman) date of March 21 to around March 11; if uncorrected, this, in turn, would cause the date of Easter (which depends on the equinox) to migrate through the seasons. Pope Gregory XIII reset the calendar by 10 days, refined the definition of the date of Easter, and changed the leap year rule to omit leap days in century years not divisible by 400. A more thorough discussion can be found in Reingold and Dershowitz [2001], which also gives extensive references.

The leap year rule for the Gregorian calendar (our present calendar) is that leap years are those divisible by 4, except not those divisible by 100, except those divisible by 400. This rule gives an average year length of $365 + 1/4 - 1/100 + 1/400 = 36597/400 = 365.2425$ days. See Shallit [1994] for a discussion of the mathematics of Gregorian-like leap year rules. The Gregorian rule more closely approximates the astronomical average in the long run, but with a more significant deviation from that average (see below).

Other calendars use different ratios and so have different leap year rules. In the Islamic calendar, a year is defined as the passage of twelve lunations (times between successive new moons) [Reingold and Dershowitz 2001]. This results in approximately $354\ \frac{11}{30}$ days per year or about 10631 days every 30 years. Since $10631 - 354 \times 30 = 11$, we need 11 leap years in every 30-year cycle. The common arithmetical approximation to the Islamic calendar has leap years in the 2nd, 5th, 7th, 10th, 13th, 16th, 18th, 21st, 24th, 26th, and 29th years of the cycle. There is an arithmetical rule for this selection but it is not obvious: the year y is a leap year if and only if $((11y + 14) \bmod 30) < 11$. Though this seems arbitrary, the formula follows from the ratio of years to days. In our presentation below, we use this arithmetical approximation to the Islamic calendar as the canonical example since it displays more of the depth and generality of the derivation than does the Julian.

The task of a calendar leap year rule is to spread the leap years as evenly as possible over the cycle of years. We assume that the physical parameters never change and that there is at least one day per year. Given a ratio of x days in c years, where x and c are positive integers, $c \leq x$, we seek an even distribution of leap years. The length in days of a normal year is $L = \lfloor x/c \rfloor$, and a leap year has $L + 1$ days. If c divides x evenly, no leap years are needed; otherwise,

$$x = cL + l,$$

where $0 < l < c$ are integers; l is the number of leap years that must occur in a cycle every c years starting with year 0. So, we insist that a year be a leap year if its number has reached or just passed an integral multiple of the ratio c/l; that is, year y is the kth leap year (with year 0 as the first leap year) if

$$(k-1)\frac{c}{l} \leq y < (k-1)\frac{c}{l} + 1, \qquad (3)$$

that is, if

$$(k-1)c \leq ly < (k-1)c + l$$

or

$$ly \bmod c < l. \qquad (4)$$

We can also generalize this by insisting that year 0 be in position s of the leap year pattern; we shift:

$$l(y + s) \bmod c < l. \qquad (5)$$

In the arithmetical approximation to the Islamic calendar,

$$11(y + 4) \bmod 30 < 11,$$

which is inequality (5) with $s = 4$. Inequality (5) also works for the cycle of

Hebrew months and years, as well as Julian, Coptic, and Ethiopic leap years (see Reingold and Dershowitz [2001]). It does *not* work for the Gregorian leap years, though it does apply to the positions of long (31-day) months versus short months on the Gregorian calendar—see Reingold and Dershowitz [2001].

The notion of "even distribution" here should be contrasted with the division into "as-equal-as-possible parts" given in Graham et al. [1994], Equation 3.24, to divide lines of text into balanced columns. Translating their division into the leap year setting, the $cL + l$ days would be divided into c years as

$$cL + l = \left\lceil \frac{cL + l}{c} \right\rceil + \left\lceil \frac{cL + l - 1}{c} \right\rceil + \cdots + \left\lceil \frac{cL + l - c + 1}{c} \right\rceil,$$

putting all the leap years at the start of the cycle. On the other hand, comparing the leap year placement of the Gregorian calendar with that of the Julian calendar—inequality (5) with $l = 1$, $c = 4$, $s = 0$—we find the year-by-year error in the Julian calendar cycles through the values 0, 1/4, 1/2, 3/4. The corresponding cycle of errors for the Gregorian calendar is 0, 97/400, $2 \times 97/400$, $3 \times 97/400$, $4 \times 97/400 - 1 = -3/100$, ...; these errors range from $-18/25$ at year 96 to 591/400 at year 303. The Gregorian wobble around its average is thus $591/400 + 18/25 = 879/400$, almost 2.2 days compared to Julian calendar's wobble of 0.75 days around its average. We want the leap years distributed as uniformly throughout the cycle so the range of errors is minimized.

So far we have just considered years. We also want to find relationships between days and years. Just as we label the years $0, 1, 2, \ldots$, so we label days $0, 1, 2, \ldots$, with day 0 being the first day of year 0.

For example, how many days are there in the range of years $0 \cdots y$? To compute this, we need to know the number of leap years k in the range of years $0 \cdots y$. For the case $s = 0$, inequality (3) can be rearranged to give

$$k = \left\lfloor \frac{y}{c/l} \right\rfloor + 1. \tag{6}$$

Using this formula, we find the number of days x in the years in the range $[0 \cdots y]$ is

$$x = L(y + 1) + \left\lfloor \frac{l(y + 1) + ((l(s - 1)) \bmod c)}{c} \right\rfloor; \tag{7}$$

this sums L days for each of the $y + 1$ years and an extra day for each leap year (see Reingold and Dershowitz [2001]). For example, the number of days in years 0, 1, 2, 3, 4 (or the number of the first day of year 5) in the arithmetical approximation to the Islamic calendar (with $s = 4$) is

$$354 \times 5 + \left\lfloor \frac{11 \times 5 + ((11 \times 3) \bmod 30)}{30} \right\rfloor = 1770 + \lfloor 58/30 \rfloor = 1771,$$

where year 2 is a leap year and both years and days are numbered starting from 0.

We can also go the other direction, finding the year at a particular day number. A derivation similar to the above is followed; the details can be found in Reingold and Dershowitz [2001], Section 1.12. We find that

$$y = y_S + \left\lfloor \frac{c(x - x_S) - ((ls - l) \bmod c) + c - 1}{cL + l} \right\rfloor, \tag{8}$$

where x_S is the day number we specify must begin year number y_S. Contrary to the convention used here, Reingold and Dershowitz [2001] require that the first day of year 1 (as opposed to year 0) be numbered day 0; this makes our functions shifted versions of theirs. For example, in Equation (8), we achieve this by letting $y_S = 1$ and $x_S = L$ (our day number for the first day of our year 1) which yields Equation (1.64) from Reingold and Dershowitz [2001].

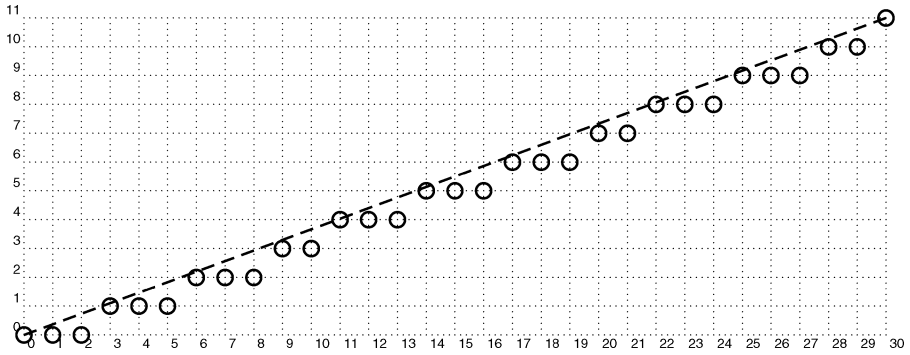


Fig. 2. Points of the leap year line for $\Delta y/\Delta x = 11/30$, $s = 0$. The jumps between steps give the leap year placement; the dashed line shows the line being approximated.

4. COMPARING THE CALCULATIONS

Let y stand for a year number and x for a day number; we want to assign y values to x values, that is, determine in what year a particular day falls. If day x falls within year y, we darken the pixel (x, y). The days are labeled sequentially, and no day is in two different years, so we have the same connectedness as Bresenham lines. A set of days in the same year forms a contiguous sequence of pixels at the same y-coordinate and the lengths of these sequences vary between two distinct lengths, L and $L + 1$, following the leap year pattern. In essence, this mapping from days to years is a set of integral lattice points approximating a line for which the slope is the exact ratio of years to days. Thus we form a correspondence of days and years with grid points and an approximation to a line by those grid points; the number $cL + l$ of days in a cycle is the run Δx, the number c of years in a cycle is the rise Δy, and the average year length $\bar{L}$ is $\Delta x/\Delta y$, the reciprocal of the slope. The number of leap years l in the run Δx is $\Delta x \bmod \Delta y$. Hence leap years can be viewed as discretizations of lines with rational slopes; we must show the Bresenham line is a special case of the leap year calculation—we do so by showing that Equation (2) is a special case of Equation (8).

Inspecting the quotients in equations (2) and (8), and using the fact that the exact average year length $\bar{L} = L + l/c$ is also the reciprocal of the slope, we can rewrite (8) in terms of Δx and Δy,

$$y = y_S + \left\lfloor \frac{\Delta y(x - x_S) - ((ls - l) \bmod \Delta y) + \Delta y - 1}{\Delta x} \right\rfloor. \tag{9}$$

We call a set of (day, year) points determined by the ratio $\Delta y/\Delta x$ and the shift s a *leap year line* (whether it actually has leap years or not). Figure 2 shows the leap year line for $\Delta y/\Delta x = 11/30$, $s = 0$. For the same ratio, a Bresenham line and a leap year line have the same pattern, but they can be shifted differently. A Bresenham line has first and last segments close to half the average line segment length, but a leap year line has arbitrary end segments depending on the shift—compare the leap year line in Figure 2 to the Bresenham line in Figure 1(b). When $s = 0$, the leap year line spreads the two type of segments (common years are segments of length L and leap years are segments of length $L + 1$) evenly.

To show now that Bresenham lines are specific cases of leap year lines, we find a leap year line whose shift s depends on the endpoints of the Bresenham line. Assuming, as we do for Bresenham lines, that the slope is between 0 and 1, we want to solve for s such that Equations (2) and (9) are equal. Comparing equations, and replacing l and c as needed, we find that

$$\Delta x = 2(\Delta y - (l(s - 1) \bmod \Delta y) - 1).$$

Table I. Summary of Comparison Between Leap Years and Bresenham Lines
(The parameters of the calendar formulas are c, L, $l < c$, and s; the parameters of the Bresenham line are x_S, y_S, x_E, and y_E.)

Calendar	Bresenham line	Calendar formula/ Bresenham formula	Approximate Islamic calendar
Day number	x-coordinate	Integer x	
Year number	y-coordinate	Integer y	
	Starting point	x_S, y_S	
	Ending point	x_E, y_E	
Years in leap year cycle	Segments in repeating pattern	$c = \Delta y = y_E - y_E$	30
Days in leap year cycle	Pixels in repeating pattern	$cL + l = \Delta x = x_E - x_S$	10631
Ratio of years to days	Slope of line	$c/(cL + l) = \Delta y/\Delta x$	30/10631
Leap years per cycle	Number of long segments	$l = \Delta x \bmod \Delta y$	11
Days in an ordinary year	Pixels in short segments	$L = \lfloor \Delta x/\Delta y \rfloor$	354
Days in a leap year	Pixels in long segments	$L + 1 = \lfloor \Delta x/\Delta y \rfloor + 1$	355
Average year length	Average segment length	$\bar{L} = (cL + l)/c = \Delta x/\Delta y$	$10631/30 = 354\frac{11}{30}$
Position of year 1 in cycle		$s = \left\lfloor \frac{2\Delta y - \Delta x - 1}{2l} \right\rfloor + 1$	4
	Initial error	Δx	
Year y is a leap year	yth segment is long	$l(y + s) \bmod c < l$	$11(y + 4) \bmod 30 < 11$
Year y begins on day x	Segment begins at (x, y)	$x = Ly + \left\lfloor \frac{ly + ((l(s-1)) \bmod c)}{c} \right\rfloor$	$x = 354y + \left\lfloor \frac{11y + 3}{30} \right\rfloor$
Day x is in year y	Point (x, y) is on the line	$y = \left\lfloor \frac{cx - ((l(s-1)) \bmod c) + c - 1}{cL + l} \right\rfloor$	$y = \left\lfloor \frac{30x + 26}{10631} \right\rfloor$

The shift s can thus be

$$s = \left\lfloor \frac{\frac{2\Delta y - \Delta x - 1}{2} + k\Delta y}{l} \right\rfloor + 1,$$

for any integer k. Choosing $k = 0$ gives

$$s = \left\lfloor \frac{2\Delta y - \Delta x - 1}{2l} \right\rfloor + 1. \qquad (10)$$

A summary of the comparisons between leap year lines and Bresenham lines is given in Table I. There are a few peculiarities that distinguish Bresenham lines and leap year lines. A Bresenham line is defined by its two endpoints; a leap year line by its slope (the ratio), the starting point, and the shift. The intention of drawing a Bresenham line is to plot a finite segment; the intention of the leap year line is to give a function for an assignment *for all days*, not just a range of days; thus, a leap year pattern repeats endlessly, but a Bresenham line does not.

5. THE UNDERLYING PATTERN

For both line drawing and leap years there are only two parameters that determine the repeating pattern: Δx and Δy (as throughout, we assume $\Delta x \geq \Delta y$). If we are only interested in these cycles, we can restrict ourselves to leap year lines with no shift, that is $s = 0$. As mentioned above with respect to Bresenham's algorithm, the pattern of alternating normal and leap segments can be difficult to see by inspection. Of course, the pattern is determined (implicitly) by equation (8), but equation (8) does not explicitly give the repeating pattern. The pattern is intimately connected with Euclid's algorithm for computing the greatest common divisor of Δx and Δy.

Euclid's algorithm can be expressed in many ways, but the most basic is given in as a function EUCLID in Algorithm 2(a), in which we use repeated subtraction to get the greatest common divisor. The proof that this subtractive algorithm is correct follows from the invariant relationship

```
   EUCLID(u, v) =
1      while u ≠ v do
           {gcd(u,v) is the greatest common
           divisor of the original parameters}
2          if u < v then
3              v := v − u
4          else
5              u := u − v
6          fi
7      od
8      return u
```

(a) Subtractive Euclid

```
   MODEUCLID(u, v) =
1      P_l := →
2      P_r := ↑
3      while u ≠ v do
           {slope(P_l) < slope(P_r),
           ||P_l||_x ||P_r||_y − ||P_r||_x ||P_l||_y = 1, and
           (u||P_l||_x + v||P_r||_x, u||P_l||_y + v||P_r||_y)
           are the original parameters}
4          if u < v then
5              v := v − u
6              P_l := P_l P_r
7          else
8              u := u − v
9              P_r := P_l P_r
10         fi
11     od
12     return (P_l P_r)^u
```

(b) Subtractive modified Euclid

Algorigthm 2. Subtractive Euclid's algorithm to compute $\gcd(u, v)$, $u, v > 0$, and a modified version to compute the leap year pattern. $\|P\|_x$ and $\|P\|_y$ are, respectively, the horizontal and vertical extents of a path P. Both algorithms take $O(u + v)$ iterations.

Table II. A Trace of MODEUCLID (30, 11) Before the Test in Line 3 is Executed (**2** is an abbreviation for →↑ and **3** is an abbreviation for →→↑. Figure 3 shows the resulting pattern)

u	v	P_l	$\|P_l\|_y/\|P_l\|_x$	P_r	$\|P_r\|_y/\|P_r\|_x$
30	11	→	0/1	↑	1/0
19	11	→	0/1	→↑	1/1
8	11	→	0/1	→→↑	1/2
8	3	**3**	1/3	**2**	1/2
5	3	**3**	1/3	**32**	2/5
2	3	**3**	1/3	**332**	3/8
2	1	**3332**	4/11	**332**	3/8
1	1	**3332**	4/11	**3332 332**	7/19
return		**3332 3332 332**	11/30		

given between lines 1 and 2, that the greatest common divisor of u and v at that point equals the greatest common divisor of the original parameters to EUCLID.

We can modify the subtractive Euclid's algorithm, so that it builds up the leap year pattern by concatenating appropriate smaller patterns; the modified algorithm is MODEUCLID in Algorithm 2(b). It composes paths P_l and P_r that correspond, respectively, to approximations from below and above the desired slope. We call MODEUCLID with the numerator and denominator of that desired slope; the initial $P_l =$ → and the initial $P_r =$ ↑ correspond to the slopes 0/1 and 1/0, where → is a rightward movement of the line of length 1 and ↑ is an upward movement. We use the notations $\|P\|_x$ and $\|P\|_y$ as, respectively, the horizontal and vertical extents of a path P; thus

$$\text{slope}(P) = \frac{\|P\|_y}{\|P\|_x}.$$

An example trace of MODEUCLID is shown in Table II. Figure 3 shows the resulting path.

The invariant relationships given between lines 3 and 4 of MODEUCLID are the heart of an inductive proof that

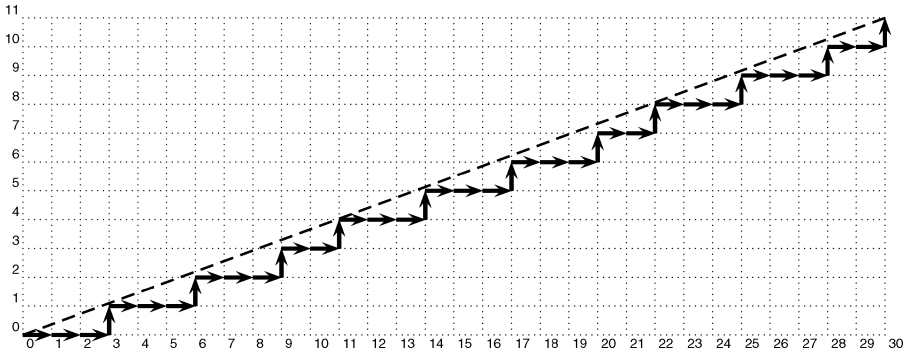


Fig. 3. The output from MODEUCLID (30, 11). It is an articulated leap year line corresponding to Figure 2. The dashed line from (0, 0) to (11, 30) touches integral lattice points only at the endpoints because gcd(11, 30) = 1.

MODEUCLID computes the correct leap year line. First, we prove that the relationships,

$$\text{original value of } u = u\|P_l\|_x + v\|P_r\|_x, \tag{11}$$

$$\text{original value of } v = u\|P_l\|_y + v\|P_r\|_y, \tag{12}$$

$$\|P_l\|_x\|P_r\|_y - \|P_r\|_x\|P_l\|_y = 1, \tag{13}$$

and

$$\text{slope}(P_l) < \text{slope}(P_r) \tag{14}$$

hold from iteration to iteration; they are clearly true at the first iteration. If $u < v$ in line 4, then we set $v := v-u$ in line 5 and $P_l := P_l P_r$ in line 6. Let us refer to values at the next moment we are between lines 3 and 4 by appending primes to them; thus we must prove that

$$\text{original value of } u = u'\|P_l'\|_x + v'\|P_r'\|_x.$$

Expressing this in terms of the old values,

$$\begin{aligned} u'\|P_l'\|_x + v'\|P_r'\|_x &= u\|P_l P_r\|_x + (v-u)\|P_r\|_x \\ &= u(\|P_l\|_x + \|P_r\|_x) \\ &\quad + (v-u)\|P_r\|_x \\ &= u\|P_l\|_x + v\|P_r\|_x, \end{aligned}$$

which, by induction (on the number of iterations), is the original value of u, as desired. Similarly, to prove that

$$\text{original value of } v = u'\|P_l'\|_y + v'\|P_r'\|_y,$$

we have

$$\begin{aligned} u'\|P_l'\|_y + v'\|P_r'\|_y &= u\|P_l P_r\|_y \\ &\quad + (v-u)\|P_r\|_y \\ &= u(\|P_l\|_y + \|P_r\|_y) \\ &\quad + (v-u)\|P_r\|_y \\ &= u\|P_l\|_y + v\|P_r\|_y, \end{aligned}$$

which, by induction, is the original value of v. Finally, to prove that (13) holds from iteration to iteration,

$$\begin{aligned} &\|P_l'\|_x\|P_r'\|_y - \|P_r'\|_x\|P_l'\|_y \\ &\quad = \|P_l P_r\|_x\|P_r\|_y - \|P_r\|_x\|P_l P_r\|_y \\ &\quad = (\|P_l\|_x + \|P_r\|_x)\|P_r\|_y - \|P_r\|_x \\ &\qquad \times (\|P_l\|_y + \|P_r\|_y) = \|P_l\|_x\|P_r\|_y \\ &\qquad - \|P_r\|_x\|P_l\|_y = 1, \end{aligned}$$

by induction. The proofs for the case $u \geq v$ are almost identical. Finally, (14) holds because (13) guarantees that

$$\text{slope}(P_l) < \text{slope}(P_l P_r) < \text{slope}(P_r).$$

Now we use the invariant relationships to prove that the path returned goes from (0, 0) to the original value of (u, v) and that it is a leap year line. The path returned, $(P_l P_r)^u$, goes from (0, 0) to

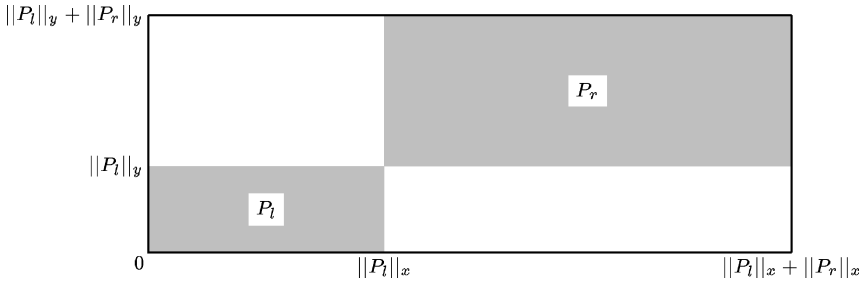


Fig. 4. If P_l is a leap year line from $(0,0)$ to $(\|P_l\|_x, \|P_l\|_y)$, P_r is a leap year line from $(0,0)$ to $(\|P_r\|_x, \|P_r\|_y)$, and $\|P_l\|_x\|P_r\|_y - \|P_r\|_x\|P_l\|_y = 1$, then the recursive structure of the leap year line from $(0,0)$ to $(\|P_l\|_x + \|P_r\|_x, \|P_l\|_y + \|P\|_y)$ is $P_l P_r$.

$(u\|P_l\|_x + u\|P_r\|_x, u\|P_l\|_y + u\|P_r\|_y)$, but because the loop ends when $u = v$,

$$\begin{aligned}&(u\|P_l\|_x + u\|P_r\|_x, u\|P_l\|_y + u\|P_r\|_y)\\ &\quad = (u\|P_l\|_x + v\|P_r\|_x, u\|P_l\|_y + v\|P_r\|_y)\\ &\quad = (\text{original value of } u,\\ &\qquad \text{original value of } v),\end{aligned}$$

by (11) and (12).

A path P is a leap year line if it passes through $(i, \lfloor i\|P\|_y/\|P_x\| \rfloor)$ for each i, $0 \le i \le \|P_x\|$. Initially, the paths P_l and P_r are (trivial) leap year lines. To prove that the path returned is a leap year line, we must show that the recursive structure shown in Figure 4 is correct—that is, that if P_l is a leap year line from $(0,0)$ to $(\|P_l\|_x, \|P_l\|_y)$, P_r is a leap year line from $(0,0)$ to $(\|P_r\|_x, \|P_r\|_y)$, and $\|P_l\|_x\|P_r\|_y - \|P_r\|_x\|P_l\|_y = 1$, then $P_l P_r$ is a leap year line from $(0,0)$ to $(\|P_l\|_x + \|P_r\|_x, \|P_l\|_y + \|P\|_y)$.

Thus to prove that $P_l P_r$ is a leap year line, we must show that it passes through $(i, \lfloor i\|P_l P_r\|_y/\|P_l P_r\|_x \rfloor)$ for each i, $0 \le i \le \|P_l P_r\|_x = \|P_l\|_x + \|P_r\|_x$, given that P_l passes through $(i, \lfloor i\|P_l\|_y/\|P_l\|_x \rfloor)$ for each i, $0 \le i \le \|P_l\|_x$, and that P_r passes through $(i, \lfloor i\|P_r\|_y/\|P_r\|_x \rfloor)$ for each i, $0 \le i \le \|P_r\|_x$. First, consider the case i, $0 \le i \le \|P_l\|_x$. It suffices to show that

$$\left\lfloor \frac{\|P_l\|_y}{\|P_l\|_x} i \right\rfloor = \left\lfloor \frac{\|P_l\|_y + \|P_r\|_y}{\|P_l\|_x + \|P_r\|_x} i \right\rfloor, \quad (15)$$

that is,

$$\lfloor \text{slope}(P_l) i \rfloor = \lfloor \text{slope}(P_l P_r) i \rfloor,$$

which would mean that the leap year line from $(0,0)$ to $(\|P_l P_r\|_x, \|P_l P_r\|_y)$ passes through the same points as P_l for $0 \le i \le \|P_l\|_x$. Equation (15) follows from proving that if $\hat{a}b - a\hat{b} = 1$ and $0 \le i < b$, then

$$\left\lfloor \frac{a}{b} i \right\rfloor = \left\lfloor \frac{a + \hat{a}}{b + \hat{b}} i \right\rfloor. \quad (16)$$

Using the identity

$$\left\lfloor \frac{a}{b} \right\rfloor = \frac{a}{b} - \frac{a \bmod b}{b},$$

proposed equation (16) becomes

$$\begin{aligned}&\frac{ai}{b} - \frac{ai \bmod b}{b}\\ &\quad = \frac{(a + \hat{a})i}{b + \hat{b}} - \frac{(a + \hat{a})i \bmod (b + \hat{b})}{b + \hat{b}},\end{aligned}$$

or

$$\begin{aligned}&\frac{(a + \hat{a})i \bmod (b + \hat{b})}{b + \hat{b}} - \frac{ai \bmod b}{b}\\ &\quad = \frac{(a + \hat{a})i}{b + \hat{b}} - \frac{ia}{b}.\end{aligned}$$

Multiplying through by $b(b + \hat{b})$ and using $\hat{a}b - a\hat{b} = 1$ transforms this into

$$\begin{aligned}&b(a + \hat{a})i \bmod b(b + \hat{b})\\ &\quad - a(b + \hat{b})i \bmod b(b + \hat{b}) = i.\end{aligned}$$

Rearranging and again using $\hat{a}b - a\hat{b} = 1$

```
    EUCLID′(u, v) =
1       while (u ≠ 0) and (v ≠ 0) do
2           if u < v then
3               v := v mod u
4           else
5               u := u mod v
6           fi
7       od
8       if u = 0 then
9           return v
10      else {v = 0}
11          return u
12      fi
```

(a) Euclid

```
    MODEUCLID′(u, v) =
1       P_l := →
2       P_r := ↑
3       while (u ≠ 0) and (v ≠ 0) do
4           if u < v then
5               v := v mod u
6               P_r := P_l^⌊u/v⌋ P_r
7           else
8               u := u mod v
9               P_l := P_l P_r^⌊u/v⌋
10          fi
11      od
13      if u = 0 then
14          return P_r^v
15      else {v = 0}
16          return P_l^u
17      fi
```

(b) Modified Euclid

Algorigthm 3. Euclid's algorithm to compute $\gcd(u, v)$, $u, v \geq 0$, and a modified version to compute the leap year pattern. The loop invariants (not shown) are identical to those in Algorithm 2. By Lamé's theorem these algorithms take $O(\log \max(u, v))$ iterations.

Table III. A Trace of MODEUCLID′(30, 11) Before the Test in Line 5 is Executed

u	v	P_l	$\|P_l\|_y/\|P_l\|_x$	P_r	$\|P_r\|_y/\|P_r\|_x$
30	11	→	0/1	↑	1/0
8	11	→	0/1	→→↑ = **2**	1/2
8	3	**3**	1/3	**2**	1/2
2	3	**3**	1/3	**332**	3/8
2	1	**3332**	4/11	**332**	3/8
0	1	**3332**	4/11	**3332 3332 332**	11/30
return		**3332 3332 332**	11/30		

makes this into

$$[a(b+\hat{b})+1]i \bmod b(b+\hat{b}) = i + a(b+\hat{b})i \bmod b(b+\hat{b}),$$

which is true provided that

$$i + a(b+\hat{b})i \bmod b(b+\hat{b}) < b(b+\hat{b}),$$

that is, if

$$a(b+\hat{b})i \bmod b(b+\hat{b}) < b(b+\hat{b}) - i.$$

Dividing this inequality by $b+\hat{b}$ transforms it to

$$ai \bmod b < b - \frac{i}{b+\hat{b}},$$

which holds because $i \leq b$ insures that $i < b+\hat{b}$. Thus (16), and hence (15), hold.

To prove that $P_l P_r$ passes through $(i, \lfloor i\|P_l P_r\|_y/\|P_l P_r\|_x \rfloor)$ for each i, $\|P_l\|_x < i \leq \|P_r\|_x + \|P_r\|_x$ is similar, but uses

$$a + \left\lfloor \frac{\hat{a}}{\hat{b}}(i-b) \right\rfloor = \left\lfloor \frac{a+\hat{a}}{b+\hat{b}} i \right\rfloor, \qquad (17)$$

given $\hat{a}b - a\hat{b} = 1$ and $b \leq i \leq b+\hat{b}$. A proof of (17) parallels our proof of (16). It follows that MODEUCLID computes the appropriate leap year line.

Of course, we do not need to do repeated subtraction in Euclid's algorithm—we can rewrite it in its more common form by using the modulus function to group repeated subtractions into a single operation. Doing so yields EUCLID′, given in Algorithm 3(a). We can similarly rewrite MODEUCLID; the result is MODEUCLID′, shown as Algorithm 3(b). Table III given the trace of MODEUCLID′

(30, 11) corresponding to that shown in Table II.

Good practice dictates that we not give an algorithm without discussing its running time. The subtractive EUCLID (and hence also MODEUCLID) take time $O(u+v)$. From Lamé's theorem we know that the worst case of the gcd-form of Euclid's algorithm is a pair of adjacent Fibonacci numbers $u = F_{i+1}$ and $v = F_i$, causing i iterations (see Knuth [1998], Theorem F, page 360). EUCLID′ thus takes $O(\log \max(u, v))$ iterations, as does MODEUCLID′.

6. CONCLUSIONS

The pattern of a Bresenham line is a special case of a leap year rule, and both are described by Euclid's algorithm, as adapted in Algorithm 3. The trace of Algorithm 3 in Table III suggests that the slopes of the paths P_l and P_r are the continuants of the continued fraction expansion of v/u; this is indeed the case. Thus our discussion can be tied to continued fractions and many other applications of Euclid's algorithm such as finding paths in the Stern-Brocot tree or finding the shortest factorization in elementary matrices of a 2×2 integer matrix with determinant 1 (see [Graham et al. 1994, Section 6.7]). Both line drawing and leap year calculations are essentially computing $\lfloor \alpha x + \beta \rfloor$, for rational α and β, and are thus also related to the Beatty sequences [Fraenkel et al. 1978].

REFERENCES

ALLOUCHE, J.-P. AND SHALLIT, J. 2003. *Automatic Sequences*. Cambridge University Press, Cambridge, U.K.

BERSTEL, J. 1990. Tracé de droites, fractions continues et morphismes itérés. In *Mots: Mélanges Offerts à M.-P. Schützenberger*, M. Lothaire, Ed. Editions Hermès, Paris, France, 298–309.

BRESENHAM, J. E. 1965. Algorithm for computer control of a digital plotter. *IBM Syst. J. 4*, 1, 25–30.

BRONS, R. 1974. Linguistic methods for the description of a straight line on a grid. *Comput. Graph. Image Process. 3*, 1, 48–62.

CASTLE, C. M. A. AND PITTEWAY, M. L. V. 1987. An efficient structural technique for encoding 'best-fit' straight lines. *Comput. J. 30*, 2, 168–175.

FRAENKEL, A. S., MUSHKIN, M., AND TASSA, U. 1978. Determination of $[n\theta]$ by its sequence of differences. *Can. Math. Bull. 21*, 441–446.

GRAHAM, R. L., KNUTH, D. E., AND PATASHNIK, O. 1994. *Concrete Mathematics*, 2nd ed. Addison-Wesley, Reading, MA.

KNUTH, D. E. 1998. *The Art of Computer Programming* (*Volume 2*: *Seminumerical Algorithms*), 3rd ed. Addison-Wesley, Reading, MA.

PITTEWAY, M. L. V. 1985. The relationship between Euclid's algorithm and run-length encoding. In *Fundamental Algorithms for Computer Graphics*, R. A. Earnshaw, Ed. Springer, Berlin, Germany, 105–111.

REINGOLD, E. M. AND DERSHOWITZ, N. 2001. *Calendrical Calculations*: *The Millennium Edition*. Cambridge University Press, Cambridge, U.K.

ROCKETT, A. M. AND SZÜSZ, P. 1992. *Continued Fractions*. World Scientific, Singapore.

SHALLIT, J. 1994. Pierce expansions and rules for the determination of leap years. *Fibonacci Quart. 32*, 5, 416–423.

SPROULL, R. F. 1982. Using program transformations to derive line-drawing algorithms. *ACM Trans. Graph. 1*, 4, 259–273.

TROESCH, A. 1998. Droites discrètes et calendriers. *Math. Inform. et Sci. Humaines 141*, 36, 11–41.

Received June 2003; accepted April 2004