

Drawing Circles, Ellipses, Parabolas and Hyperbolas in Logo/W 1.0

Pavel Boychev
University of Sofia
Faculty of Mathematics and Informatics,
5, James Bourchier Str.,
Sofia 1126
BULGARIA
e-mail: bojo@fmi.uni-sofia.bg

Abstract

Few computer applications, which fall into the class of graphical applications, support curve drawing. Of course we can exclude several specialised applications that are designed just for drawing such curves. All the rest applications can draw circles and a limited class of ellipses while parabolas and hyperbolas must be explicitly programmed.

The Logo/W 1.0 system is designed to allow creating models of many different objects and phenomena from various fields of knowledge. Examples of such models are Algebra, Planimetry, Optics, Logic and so on. That leads to the need of defining, drawing and using circles, ellipses, parabolas and hyperbolas. Such geometry curves are significantly used in Planimetry and Optics. The current material will focus on drawing these curves and the arising problems that have to be resolved in order to draw them quickly and accurately.

Clipping

One of the basic problem while visualising curves is that the drawing process is not as fast as the one of a line. Any part of a line is similar to any another part. That fact is not applicable to circles, ellipses, parabolas and hyperbolas. The slower drawing of curves (compared to lines) is more obvious when the curve is situated in such manner that only a small part of it visible. The used resources (mainly memory and time) are spread over the whole curve, not only over the visible part.

To eliminate and to bypass unnecessary calculations applications use **clipping**. Clipping is usually a separate step in drawing curves during which is determined what part of it is visible. Further calculations are applied only on the visible part of the curve.

There are two kinds of clipping but both of them are not suitable for drawing circles, ellipses, parabolas and hyperbolas. The first kind of clipping is to let the basic graphical environment do this. Logo/W 1.0 runs under Windows 3.1 or Windows 95 therefore the basic graphical environment is implemented in the form of GUI (Graphical User Interface). If Windows is responsible for clipping curves, that could lead to noticeable limitations. GUI drawing primitives (i.e. drawing separate pixels or lines) are based on 2-byte-integer-parameter convention. That means that the range of values [-32768..32767] limits the

drawable size of circles and ellipses. Unfortunately any parabola or hyperbola consist of infinite number of points placed beyond that range.

The second kind of clipping is to calculate intersection points between the curve and the border of the visible area of the screen. Usually that area is a rectangle with sides parallel to the coordinate axes. In the current material we will consider visible area is just of that type.

To implement the second kind of clipping, a curve must be represented by parametric equation. If the point $P(x,y)$ is a point of a curve, there must exist a value t that satisfies the equations:

$$(1) \quad \begin{cases} X = X(t) \\ Y = Y(t) \end{cases}$$

Clipping the curve is equivalent of solving the following equations:

$$(2) \quad \begin{cases} \text{Min}X \leq X(t) \leq \text{Max}X \\ \text{Min}Y \leq Y(t) \leq \text{Max}Y \end{cases}$$

The problems of solving inequations (2) is that for curves they include terms like $a + bt + ct^2$ or $\sqrt{a + bt + ct^2}$ that are hard enough to solve quickly and precisely. The common solution consist of eight values, thus forming four intervals. It is not so obvious how to combine these values into the intervals. Moreover, some values may be the same (forming an empty interval) or even, there may be less solutions.

Suppose we have already calculated the correct intervals. Drawing the visible part of the curves is simple enough and will not be discussed in details.

Logo/W 1.0 Clipping

As already stated above, both kinds of clipping are improper for drawing circles, ellipses, parabolas and hyperbolas. In order to bypass all the problems a new approach is designed and implemented into Logo/W 1.0. Clipping is a two-step process combining mathematical and GUI clipping together.

The first step is to solve the inequations:

$$(3) \quad \begin{cases} \text{Min}X \leq \bar{X}(t) \\ \text{Max}X \geq \underline{X}(t) \\ \text{Min}Y \leq \bar{Y}(t) \\ \text{Max}Y \geq \underline{Y}(t) \end{cases}$$

where

$$(4) \quad \underline{X}(t) \leq X(t) \leq \bar{X}(t) \text{ , and}$$

$$(5) \quad \underline{Y}(t) \leq Y(t) \leq \bar{Y}(t)$$

The functions $\underline{X}(t)$, $\bar{X}(t)$, $\underline{Y}(t)$ and $\bar{Y}(t)$ are linear function, thus allowing an easy and fast solving of system (3). It is very important to find out as restrictive linear functions satisfying (4) and (5) as possible. More restrictive functions lead to more accurate clipping. It

is important to note that inequations (3) are used to clip and exclude that parts of the curves that are close or beyond then 32768 limit.

The second step is to draw the curves using the intervals calculated from (3). That intervals include all the visible part of the curves and parts of the invisible. The final clipping is performed by GUI. There will be no problems, because during step 1 all extremely big values (above 30000 and below -30000) are eliminated.

Linear Restriction

Designing proper linear function is very important in order efficiently to eliminate big values. The fact is that a usable restricting linear function may be constructing only by dividing the curve into several parts and for each part a different set of functions is used. The mostly used intervals are $[-\infty, -1]$, $[-1, 0]$, $[0, 1]$ and $[1, \infty]$. These intervals are defined according to the inequations to be solved and are also affected by the chosen curve parametric representations. When restricting functions are based on trigonometric sine and cosine, the intervals are $[0, \frac{1}{2}\pi]$ and $[\frac{1}{2}\pi, \pi]$.

If another restricting linear functions then the implemented ones in Logo/W 1.0 are selected, then probably another intervals should be more suitable.

Common Curve Characteristics

To reveal the characteristics of curves must be cleared the question what equations are used. Below are listed a little bit normalised curve equations:

$$(6) \quad (x - x_0)^2 + (y - y_0)^2 = r^2 \quad \text{for circles}$$

$$(7) \quad \frac{(x - x_0)^2}{a^2} + \frac{(y - y_0)^2}{b^2} = 1 \quad \text{for ellipses}$$

$$(8) \quad y - y_0 = p(x - x_0)^2 \quad \text{for parabolas}$$

$$(9) \quad \frac{(x - x_0)^2}{a^2} - \frac{(y - y_0)^2}{b^2} = 1 \quad \text{for hyperbolas}$$

Equations (6) - (9) are not the parametric equations that are used when drawing curves. Instead they are used only as explanation of geometric meaning of common curve characteristics.

All curves have several common characteristics: center, angle of rotation, radius and precision. Every curve have a central point called shortly **center**. That point determines the offset of the curve from the origin of the coordinate system. For parabolas the central point is that at its vertex. For circles, ellipses and hyperbolas the central point corresponds to the geometrical center. Formally the center of a curve is the point $P(x_0, y_0)$ where x_0 and y_0

are from (6) - (9). **Angel of rotation** is a parameter that indicates the rotation of the curve. Only circles are not affected by the angel of rotation therefore they have not got such a characteristic. Equations (6) - (9) are given at angle of rotation $\phi = 0$. **Radius** is another characteristic of curves. For circles its is the same as the geometric radius and corresponds to r in (6). For ellipses (7) and hyperbolas (9) it is a pair of the two radiuses: (a,b). Parabolas have no radius at all. For such curves (8) this characteristic corresponds to the parameter p . **Precision** is not a geometric characteristic. It defines how smooth is the appearance of a curve. Precision is defined by a number of points. Every curve is divided into four separate parts, called *subcurves*. Each subcurve is clipped with inequations (3). The remaining subcurves (if any) are divided into the number of points declared by the precision. So, if precision is N then for each curve there are nearly $4N$ points calculated. The curve is drawn as a set of segments connecting each N points into a separate path. Every subcurve is drawn as a single path.

Transforming To Linear Equations

To demonstrate how complex inequations are transformed into linear ones we will discuss in details two different examples. The first is solving the inequation:

$$(10) \quad At + B\frac{1}{t} \geq C$$

Clipping hyperbolas in Logo/W 1.0 is done with a set of inequations like (10) where A , B and C are real numbers (positive, negative or zero - if there is \leq sign instead of \geq , the coefficients A , B and C are negated). Transforming into linear inequation depends on the range of t . Let us examine the case $t \in [1, \infty)$. If $B = 0$ then (10) is equivalent to (11):

$$(11) \quad At \geq C$$

If $B > 0$ we can replace $B\frac{1}{t}$ with B because $B \geq B\frac{1}{t}$. Thus (10) is transformed to inequation (12) which is not so restrictive, but have the incredible property to be linear.

$$(12) \quad At + B \geq C$$

if $B < 0$ we can replace $B\frac{1}{t}$ with 0 because $0 \geq B\frac{1}{t}$. The situation is the same as in the case of $B=0$, so inequation (10) is transformed to (11). In an appropriate way is processed the case $t \in (0,1)$ with the only difference that transformed is At rather than $B\frac{1}{t}$.

The second example of how complex inequations are transformed into linear ones is used in drawing ellipses. The problem is how to make linear restriction of the trigonometric functions sine and cosine. Let us use the following equation (13) as example:

$$(13) \quad A \sin \theta + B \cos \theta + C \geq 0$$

For $\theta \in [0, \frac{1}{2}\pi]$ sine and cosine are restricted by the next two inequations:

$$(14) \quad \underline{\sin} \theta = \frac{2}{\pi} \theta \leq \sin \theta \leq \theta = \overline{\sin} \theta$$

$$(15) \quad \underline{\cos} \theta = 1 - \frac{2}{\pi} \theta \leq \cos \theta \leq \theta - \frac{\pi}{2} = \overline{\cos} \theta$$

For another interval $\theta \in [\frac{1}{2}\pi, \pi]$ equations (14) and (15) would be different. Let us return to (13). According to the signs of A and B we choose what linear restriction of sine and cosine are selected. If $A < 0$ and $B > 0$ sine must be replaced by $\underline{\sin}$, and cosine by $\overline{\cos}$. In this case equation (13) is transformed to (16):

$$(16) \quad \frac{2A}{\pi} \theta + B \left(\theta - \frac{\pi}{2} \right) + C \geq 0 \quad \Leftrightarrow \left(\frac{2A}{\pi} + B \right) \theta + C - \frac{B\pi}{2} \geq 0$$

Performance Test

Following the above formulas a set of drawing programs are designed. The programs are capable of visualising circles, ellipses, parabolas and hyperbolas. To test how fast these curves are drawn a special performance test has been evaluated. Circles are denoted by $\sigma(x_0, y_0, R)$, ellipses by $\epsilon(x_0, y_0, \phi, a, b)$, parabolas by $\pi(x_0, y_0, \phi, p)$ and hyperbolas by $\chi(x_0, y_0, \phi, a, b)$.

The test environment includes IBM PC Compatible, 80486 DX2 / 66 MHz, 8 MB RAM, Windows 3.10, Borland Pascal for Windows 7.0.

	Circles $\sigma(x_0, y_0, R)$	Ellipses $\epsilon(x_0, y_0, \phi, a, b)$	Parabolas $\pi(x_0, y_0, \phi, p)$	Hyperbolas $\chi(x_0, y_0, \phi, a, b)$
Test conditions $T \in [1, 10^4]$	$x_0 = y_0 = 0$ $R = 10 + \frac{T}{20}$	$x_0 = y_0 = 0$ $\phi = T$ $a = 10 + \frac{T}{20}$ $b = 10 + \frac{T}{50}$	$x_0 = y_0 = 0$ $\phi = T$ $p = 10 + \frac{T}{20}$	$x_0 = y_0 = 0$ $\phi = T$ $a = 10 + \frac{T}{20}$ $b = 10 + \frac{T}{50}$
Total execution time (sec.)	55.8 sec	85.6 sec	42.6 sec	48.0 sec
Seconds per curve	0.006 sec	0.009 sec	0.004 sec	0.005 sec
Curves per second	179 circles	117 ellipses	235 parabolas	208 hyperbolas

All tests are performed for curves separated exactly by 40 points (i.e. $N \approx 10$).

Parabolas and hyperbolas have a better score, because their drawing process is optimised concerning used mathematical operations. After a suitable optimisation, circles and ellipses could score up to 200 curves per second.

Source Listing

Below is a listing of a full-featured standalone source code for drawing hyperbolas. To run it you must have Windows 3.1 and Borland Pascal 7.0. With minor changes it can be adapted to any other platform. The source code for drawing circles, ellipses and parabolas is quite similar.

```

program Hyperbola;
uses OWindows, WinTypes, WinProcs;
const MaxX = 200;
        MinX = -350;
        MaxY = 200;
        MinY = -100;
        N    = 9;

var MyApp : TApplication;
    DC     : HDC;

procedure Hyp(DC:HDC; A,B,X0,Y0,F:real);
  {-Draws hyperbola}
  var T1,T2,TS : real;
      S,C,AS,AC,BS,BC : real;
      K1,K2,K3,K4,KB1,KB3,KS1,KS3 : real;
      I    : integer;
      H,V  : longint;

  function Solve(K1,K2,M:real):boolean;
    {-Solves the equation  $K1*t+K2/t=M$ }
    begin
      Solve:=True;
      if K1>0 then M:=M-K1;
      if K2=0
        then begin if M>0 then exit; end
        else if K2<0
          then if M>=0
            then exit
            else begin
              M:=K2/M;
              if T1<M then T1:=M;
            end
          else if M>0
            then begin
              M:=K2/M;
              if T2>M then T2:=M;
            end;
        end;
      Solve:=False;
    end; {-Solve}

```

```

procedure Draw;
  {-Draws a one forth of a hyperbola}
  var I:integer;
  begin
    K1:=AC+BS;
    K2:=AC-BS;
    K3:=-AS+BC;
    K4:=-AS-BC;

    T1:=0;
    T2:=1;
    if Solve(K1,K2,MinX-X0) then exit;
    if Solve(-K1,-K2,-MaxX+X0) then exit;
    if Solve(K3,K4,MinY-Y0) then exit;
    if Solve(-K3,-K4,-MaxY+Y0) then exit;
    if T1>T2 then exit;

    TS:=(T2-T1)/Pred(N);
    if T1=0 then T1:=T1+TS;
    KB1:=K1*T1; KS1:=K1*TS;
    KB3:=K3*T1; KS3:=K3*TS;
    for I:=1 to Succ(N) do
      begin
        H:=Round(400+(KB1+K2/T1+X0));
        V:=Round(300-(KB3+K4/T1+Y0));
        if I=1
          then MoveTo(DC,H,V)
          else LineTo(DC,H,V);
        T1:=T1+TS;
        KB1:=KB1+KS1;
        KB3:=KB3+KS3;
      end; {-for}
    end; {-Draw}

  begin {-Hyp}
    F:=F*Pi/180;
    S:=Sin(F);
    C:=Cos(F);
    A:=A/2;
    B:=B/2;

    AC:=A*C; AS:=A*S; BC:=B*C; BS:=B*S;
    Draw;
    S:=-S; C:=-C; AS:=-AS; AC:=-AC; BS:=-BS; BC:=-BC;
    Draw;
    B:=-B; BS:=-BS; BC:=-BC;
    Draw;
    S:=-S; C:=-C; AS:=-AS; AC:=-AC; BS:=-BS; BC:=-BC;
    Draw;
  end; {-Hyp}

  begin {-Main program}
    MyApp.Init('');
    MyApp.MainWindow^.Show(sw_ShowMaximized);

    DC:=GetDC(MyApp.MainWindow^.HWindow);
    Rectangle(DC,MinX+400,300-MinY,MaxX+400,300-MaxY);
    Hyp(DC,120,80,10,-10,30);
    ReleaseDC(MyApp.MainWindow^.HWindow,DC);

    MyApp.Done;
  end. {-Main program}

```

References

1. Rogers, D., F., (1985), *Procedural Elements for Computer Graphics*, McGraw-Hill Book Company.
2. Nõàíèëíâ, Ã., Áíðèñíâ, À., (1988), Ííâè ñðáùè ñ êííè-íèðà ñâ-áíèÿ, Íàðíáíà Íðíñâáðà, Ñíòèÿ.
3. Êíââ-ââ, Ì., (1995), Ãðàðè-áí èíðàððáéñ íà ííââðà Microsoft Windows ðààèèçàòèÿ íà Ñèñòáìà Ìèàíèìáððèÿ, äèèëííá ðàáíðà, ÑÓ "ñâ. Êè. Íððèññèè", ÕÌÈ, Ñíòèÿ

Biography

Pavel Boychev was born in 1969 in Stara Zagora . He graduated from the Faculty of Mathematics and Informatics at Sofia University in 1995. Since 1991 he has been working in the Department of Information Technologies as a programmer. He is actively involved in the designing, developing and advancing logo-based programming environment and language extensions.