The Midpoint Ellipse Algorithm

Kenneth H. Carpenter
Department of Electrical and Computer Engineering
Kansas State University

February 10, 1994

1 Introduction

The text, *Introduction to Computer Graphics*, by Foley, *et. al.*, derives an algorithm for scan conversion of a circle having an integer radius with center at the origin by use of a midpoint decision criterion. This approach may be extended to scan convert an ellipse having its center at the origin, integer values for major and minor axes, and its axes parallel to the coordinate axes. The derivation of the algorithm follows.

2 Derivation

2.1 Basic algorithm

An ellipse with center at the origin and axes parallel to the $x - y$ axes has the formula

$$f(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0, \tag{1}$$

where a and b are the radii, and where f is positive if a point (x, y) lies outside the ellipse, is negative if a point is inside the ellipse, and is zero if a point is on the ellipse. The slope of the ellipse is given at any point on it by

$$\frac{dy}{dx} = -\frac{\partial f}{\partial x} \Big/ \frac{\partial f}{\partial y} = -\frac{b^2 x}{a^2 y}. \tag{2}$$

Suppose there is a function EllipsePoints that not only causes the pixel at (x, y) to be written, but also pixels at $(-x, y)$, $(-x, -y)$, and $(x, -y)$. Then this function can be called to complete the ellipse while only one-fourth of it is scanned.

We begin by starting at $(0, b)$ at the top of the ellipse. The slope will be less than one in magnitude initially, so the variable to increment will be x. At the new x one uses f to determine whether or not to increment y. This continues until one reaches the location where the slope is greater than one in magnitude. Then x and y change rolls, and one continues to $y = 0$. The algorithm is then:

```
while ‘‘slope magnitude’’ less than one:
    increment x;
    if f(x,y-.5)>0 then decrement y and update f;
    else update f;
    EllipsePoints(x,y)
end while

while y > 0:
    decrement y;
```

```
        if f(x+.5,y)>0 then update f;
        else increment x and update f;
        EllipsePoints(x,y)
    end while
```

(One must also set the four points at the ends of the axes.)

2.2 Incremental implementation

The key to an efficient algorithm is being able to keep the needed value of the decision function (proportional to) f at its current value by simple arithmetic operations done incrementally, rather than recalculating its value at each point. For the equation of the ellipse, to gain much efficiency, one needs to go to second order differences.

To keep integer values for all quantities, multiply eq.(1) by $4a^2b^2$ to get a new function that has the same properties with regard to the curve.

$$h(x, y) = 4b^2x^2 + 4a^2y^2 - 4a^2b^2 \tag{3}$$

Now $h = 0$ defines the ellipse, $h > 0$ gives points outside the ellipse and $h < 0$ gives points inside the ellipse.

The starting value needed for h is at $(1, b - \frac{1}{2})$. This value is the initialization for the decision variable:

$$h(1, b - 0.5) = 4b^2 + a^2(1 - 4b). \tag{4}$$

When incrementing the decision variable one makes use of the formula

$$h(x + dx, y + dy) = h(x, y) + 4b^2dx(2x + dx) + 4a^2dy(2y + dy). \tag{5}$$

At each step, x is incremented by zero or one and y is incremented by zero or minus one. Thus the increment to be added to the decision variable is one of

$$\begin{aligned} d1 &= 4b^2(2x + 1) \text{ for x only incremented} \\ d2 &= -4a^2(2y - 1) \text{ for y only decremented} \\ d3 &= d1 + d2 \text{ for both x incremented and y decremented.} \end{aligned} \tag{6}$$

For efficiency of calculation, to replace multiplies by adds, on needs to use an incremental method of generating the $d1$ and $d2$ values. Thus the initial values for $(x, y) = (1, b - 0.5)$ are

$$\begin{aligned} d1 &= 12b^2 \quad \text{and} \\ d2 &= -8a^2(b - 1) \end{aligned} \tag{7}$$

and on steps of one in the x direction, $d1$ is incremented by adding $8b^2$ while on steps of minus one in the y direction $d2$ is changed by adding $8a^2$.

The test for magnitude of slope less than one may be expressed as a test for $b^2x < a^2y$ The values of b^2x and a^2y could be maintained by incrementing rather than calculation at each step, if additions are much faster than multiplications. The needed increments are obvious.

New initial values are needed for h, d1, and d2 when beginning the segment for which $b^2x > a^2y$. As an alternative to writing separate code for this segment, the code for the segment beginning at $x = 0$ could be reused with x and y interchanged and a and b interchanged.

These rules may then be implemented in a computer code to scan convert the ellipse. If the radii are too large, the the values for h, d1, and d2 may overflow integer values. In that case a floating point implementation must be used, and the restriction of the radii and center to integer values might as well be removed by suitable modification to the algorithm.

```
/*midellip.c - K.H.Carpenter - khc@eece.ksu.edu*/
/*Last changed 11FEB94*/
/*midellip is a function to scan convert an ellipse given
 *its major and minor radii.  These radii must be integers,
 *and must be parallel to the coordinate axes.  Points are
 *set by call to an external function EllipsePoints, which
 *must do any translation of the center of the ellipse from
 *the origin.  The four pixels at the ends of the axes are
 *not set by this routine and must be handled separately.
 *
 *REFERENCE: Kenneth H. Carpenter, "The Midpoint Ellipse Algorithm,"
 *February 10, 1994, and, {\em Introduction to Computer Graphics},
 *by Foley, Van Dam, Feiner, Hughes, and Phillips, Addison-Wesley,
 *1993, section 3.3.
 */

/*ENTRYS:
 ****************************
 *  void midellip(int a, int b)
 *
 *  Purpose:
 *      Scan converts the ellipse with major and minor
 *      radii a,b by calling an external routine to set
 *      the points a x,y values and their symmetrical
 *      equivalents, with x,y assumed to have their orign
 *      at the ellipse center.
 *
 *  Input arguments:
 *      a - the horizontal (x) radius of the ellipse
 *      b - the horizontal (y) radius of the ellipse
 *
 *  Error handling:
 *      none - caller must ensure that a,b are both positive
 *
 *  Prototype in:
 *      none available
 */

/*EXTERNALS:
 *  Functions called:
 *      void EllipsePoints(int x, int y)
 *      standard c library
 *
 *  No header files included
 */

/*PORTABILITY:
 *  Uses standard c with ANSI prototypes
 */

/*REVISION HISTORY:
 *10FEB94 - initial version
 *11FEB94 - added conditional for 64 bit integers
 */
```

```
/*************************************************************/
/*includes for file:*/
#include <stdio.h>
/*definitions for file:*/
extern void EllipsePoints(int x, int y);

  /*For a system with only 16 bit integers, INT needs
   *to be defined as long to get 32 bit integers to allow
   *for the squarings that occur.
   *For resolutions that would have radii on the order of
   *1000, then even 32 bit integers are not large enough.
   *If type "double long" exists (as in gcc) it could be used,
   *but the execution time would likely be as long as if
   *a floating point algorithm were used.
   *With double long types, gcc does not allow use of binary shift
   *operator.
   */
#ifdef DLONG
#define INT double long
#else
#define INT int
#endif

/*************************************************************/
void midellip(int a, int b){
    int x = 0, y = b;   /*pixel coordinates*/
#ifndef DLONG
    INT a2 = a*a, b2 = b*b; /*squares of axes*/
    INT h, d1, d2, sn, sd;  /*decision variables and increments*/
    INT a2t8 = a2 << 3, b2t8 = b2 << 3;
#else
    INT a2, b2; /*squares of axes*/
    INT h, d1, d2, sn, sd;  /*decision variables and increments*/
    INT a2t8, b2t8;
    a2 = a*a;
    b2 = b*b;
    a2t8 = a2*8;
    b2t8 = b2*8;
#endif

    /*initialize for first midpoint at x=1, y=b-0.5 */
#ifndef DLONG
    h = (b2t8 >> 1) + a2*(1 - (b << 2));  /*use shift for multiply by 2*/
#else
    h = b2*4 + a2*(1 - 4*b);
#endif
    d1 = 12*b2;
    d2 = -a2t8*(b - 1);
#ifdef DEBUG
printf("a2t8=%ld\n",a2t8);
printf("a=%d, b=%d, h=%ld, d1=%ld, d2=%ld\n",a,b,h,d1,d2);
#endif
    sn = b2;
```

```
#ifdef DLONG
    sd = (a2*b) - (a2/2);   /*truncation error here won't affect decision*/
#else
    sd = (a2*b) - (a2 >> 1);   /*truncation error here won't affect decision*/
#endif

    /*begin loop for incrementing x*/
    while(sn < sd){ /*magnitude of slope less than one*/
#ifdef DEBUG
printf("d1 = %d, d2 = %d, h = %d\n",d1,d2,h);
#endif
        if(h > 0){  /*midpoint is outside ellipse so decrement y*/
            y--;
            h += d2;
            sd -= a2;
            d2 += a2t8; /*must be changed after use in changing h*/
        }
        x++;    /*always increment x and the following values*/
        h += d1;
        sn += b2;
        d1 += b2t8;
#ifdef DEBUG
printf("sn = %ld, sd = %ld\n",sn,sd);
#endif
        EllipsePoints(x,y);  /*set the pixels*/
    }/*end of while on slope magnitude less than one*/

/*Here, before continuing with decrementing on y, must initialize
 *the decision variable for the new midpoints by changing the values
 *of h, d1, and d2 for x greater by 0.5 and y less by 0.5.
 *
 *From the formulas for h, the amount to be added to h is
 * -b^2(4x + 3) - a^2(4y - 3).
 *The amounts to change d1 and d2 are 4b^2 and 4a^2.
 */
#ifndef DLONG
    /*These formulas add increments to convert to the new starting pt.*/
    h -= a2*((y << 2) - 3) + b2*((x << 2) + 3);
    d1 -= (b2 << 1);
    d2 += (a2 << 1);
#else
    /*These formulas recalculate the values for the new starting point.*/
    h = b2*(4*x*x + 4*x +1) + 4*a2*(y-1)*(y-1) - 4*a2*b2;
    d1 = b2t8*(x+1);
    d2 = -4*a2*(2*y - 3);
#endif
    while(y > 1){   /*continue with decrementing y*/
#ifdef DEBUG
printf("d1 = %d, d2 = %d, h = %d\n",d1,d2,h);
#endif
        if(h < 0){  /*midpoint is inside ellipse so increment x*/
            x++;
            h += d1;
            d1 += b2t8;
```

```
        }
        y--;    /*always decrement y, h,  and d2*/
        h += d2;
        d2 += a2t8;

        EllipsePoints(x,y);
    }
}

#ifdef TEST
#include <stdio.h>
void main(int arc, char **argv){
    int a,b;    /*assume they are the first two arguments*/
    a = atoi(argv[1]);
    b = atoi(argv[2]);
    midellip(a,b);
}
void EllipsePoints(int x, int y){
    printf("%d,  %d\n",x,y);
}
#endif
```