Metaprogramming for the Generation of Nonparametric Curves

V. Boyer

G.R.I.S - Laboratoire d'Intelligence Artificielle
Université Paris 8
2, rue de la Liberté
93526 Saint-Denis Cedex - FRANCE
boyer@ai.univ-paris8.fr

Abstract

One of the most important functions of paintboxes is drawing curves. These primitives have been programmed and the user can never add a new program which computes the discrete points of a given function. Using metaprogramming and the Jordan's method, our program CAPC automatically generates, for a given function, a new program which computes the discrete points for this function and adds it to our paintbox tool.

1. Introduction

One of the most important functions of paintboxes is drawing curves such as lines, circles, elliptic arcs... For each primitive, an algorithm has been written[1, 2, 3] generally based on the DDA method. In 1973 Jordan et al.[4] proposed an algorithm for the generation of nonparametric curves which displayed the points of the discrete plane closer to the continuous curve.

We propose an efficient extension of this method. We have created a program named CAPC which generates for a nonparametric curve the corresponding program code and the binary file using metaprogramming and the Jordan's algorithm.

2. The Jordan's algorithm

Consider a two-dimensional curve described by $f_r(x,y)=0$. Let P and Q be two discrete points of the curve. P is named the starting point and Q the ending point. The Jordan et al.[4] method allows to generate an algorithm which determines the discrete points between P and Q closer to the curve using a step size.

The computation of the derivates is necessary. We will assume that the function has continuous derivates. In particular we use the notation:

$$f_x=\frac{\partial f_r}{\partial x},\ f_y=\frac{\partial f_r}{\partial y},\ f_{xx}=\frac{\partial^2 f_r}{\partial x^2},\ f_{xy}=\frac{\partial^2 f_r}{\partial x\partial y},\ f_{yy}=\frac{\partial^2 f_r}{\partial y^2}$$

The purpose of the algorithm is to produce a set of points that will "follow" the curve as closely as possible. Starting with P there are eight points to which we can step and two direction of trace along the curve: $+\vec{v}=(-f_y,\ f_x)$ and $-\vec{v}$. An integer D is used to choose the direction.

$$D=0 \quad \text{if} \quad -\vec{v} \quad \text{is chose,}$$
$$D=1 \quad \text{if} \quad +\vec{v} \quad \text{is chose.}$$

Thus by choosing D, the eight possible points to which we can step are reduced to three. Let the point (x,y) be a point of the discrete curve closer to f_r then the three possible points are:

$$(x+Inc_x,y),\ (x+Inc_x,y+Inc_y) \text{ and } (x,y+Inc_y)$$

Inc_x and Inc_y are two integer values in $\{-1,1\}$.

$$Inc_x=\begin{cases}+1, & \text{if } ((f_y\geq 0) \text{ and } (D=0)) \text{ or } ((f_y<0) \text{ and } (D=1))\\ -1, & \text{if } ((f_y<0) \text{ and } (D=0)) \text{ or } ((f_y\geq 0) \text{ and } (D=1))\end{cases}$$

$$Inc_y=\begin{cases}+1, & \text{if } ((f_x<0) \text{ and } (D=0)) \text{ or } ((f_x\geq 0) \text{ and } (D=1))\\ -1, & \text{if } ((f_x\geq 0) \text{ and } (D=0)) \text{ or } ((f_x<0) \text{ and } (D=1))\end{cases}$$

So, at any point it is only necessary to choose be-

tween three points. We consider $f^x = f_r(x+Inc_x, y)$, $f^y = f_r(x, y+Inc_y)$ and $f^{xy} = f_r(x+Inc_x, y+Inc_y)$.

It is only necessary to compare $|f^x|$, $|f^y|$ and $|f^{xy}|$ to decide which point to choose.

$|f^x| < |f^y|$ and $|f^x| < |f^{xy}|$ the next point is $(x+Inc_x, y)$
$|f^y| < |f^x|$ and $|f^y| < |f^{xy}|$ the next point is $(x, y+Inc_y)$
$|f^{xy}| < |f^x|$ and $|f^{xy}| < |f^y|$ the next point is $(x+Inc_x, y+Inc_y)$

Then the variables and the partial derivates are easily updated. The final expression is now presented. Letting $\alpha = x$ if a step in x were taken, $\alpha = y$ if a step in y were taken, and $\alpha = xy$ if a step in x and y were taken, the decision variables can be updated:

$$f^x = f^\alpha + f_x Inc_x + \frac{1}{2} f_{xx} (Inc_x)^2 + \dots$$

$$f^y = f^\alpha + f_y Inc_y + \frac{1}{2} f_{yy} (Inc_y)^2 + \dots$$

$$f^{xy} = f^\alpha + f_x Inc_x + f_y Inc_y + \frac{1}{2}(f_{xx}(Inc_x)^2 + 2 f_{xy}(Inc_x)(Inc_y) + \dots$$

The figure 1 presents the Jordan's algorithm. This algorithm produces for a given nonparametric curve a set of points closer to the continuous curve. The programmers must calculate the derivates and program it in the *Init* function. The function named *Incremental* updates the derivatives through an incremental computation.

```
courbe(int x_p, int y_p, int x_q, int y_q) {
  int x = xp ;
  int y = yp ;
  int f^α;
  Writepixel(x, y) ;
  Init(f_x, f_y, f_xy, D);
  while (x != x_q  &&  y != y_q) {
    Compute(Inc_x, Inc_y, f^x, f^y, f^xy);
    if (abs(f^x) <= abs(f^y)) {
      if (abs(f^x) <= abs(f^xy)) {
        Inc_y = 0 ;
        f^α = f^x ;
      }
      else {
        f^α = f^xy ;
      }
    }
    else {
      if (abs(f^y) <= abs(f^xy)) {
        Inc_x = 0 ;
        f^α = f^y ;
      }
      else {
        f^α = f^xy ;
      }
    }
    x += Inc_x;
    y += Inc_y;
    Writepixel(x, y);
    Incremental(f_x, f_y, f_xy);
  }
}
```

Figure 1: *Jordan's algorithm.*

3. Metaprogramming

We have adapted the Jordan's method to generate automatically the algorithms. The user gives a function f_r and our program named CAPC writes a new program which computes the points of the discrete curve closer to f_r.

We consider a two-dimensional curve $f_r(x, y)$. Parameters can be used in the definition of f_r. These parameters must be small letters. Only x and y are reserved and can not be considered as parameters.

To obtain a correct formulation without ambiguity the user must write the function using the **C**-like language:

- $f_r(x, y) = a*x*x - y$
- $f_r(x, y) = a*x + b*y + c$
- $f_r(x, y) = x*x + y*y - r*r$

We describe now the process which allows to choose a point of the discrete curve. The process is the same as Jordan's and consists in:

1. computing the derivates in x and y,
2. computing Inc_x and Inc_y through D,
3. computing f^x, f^y and f^{xy} for a given point,
4. determining the value closest to zero.

This process can be iterated for each point of the discrete curve.

All programs generated by CAPC have the same structure. The only difference between two of these programs is the function f_r and the derivates in x and y. We have written a program named *source* in **C**, which uses three macro-definitions (#define DERIVATEX, #define DERIVATEY, #define FUNCTION). CAPC performs the following operations:

1. a lexical analysis of f_r,
2. the computation of f_x and f_y,
3. the copy of the program *source* into a new file. The name of which is specified by the user,
4. the creation of a header file where the macro-definitions the parameters used in f_r are defined,
5. the compilation of the new generated program.

We obtain automatically a binary file. The parameters are a starting point, an ending point, a direction and the values of the parameters of f_r.

4. Results

We present the results for the three functions (see Alg 2, 3 and 4). We present only the header file. These three files have been automatically generated.

5. Conclusion

We have presented CAPC . It generates automatically, by metaprogramming, program which computes the points of the discrete curve closer to f_r. This is an extension of the Jordan's method. CAPC has been included in a paintbox. A large collection of programs generated by CAPC exists.

We will extend this method to the generation of nonparametric curve with three variables. This extension will allow to generate automatically shading supports for the Bourdin color shading method[5].

References

1. J.E. Bresenham. Algorithm for computer control of a digital plotter. *IBM System Journal*, 4(1):25–30, 1965. 1
2. J.E. Bresenham. A Linear Algorithm for Incremental Digital Display of Circular Arcs. *Communications of the ACM*, 20(2), 1977. 1
3. M.L.V. Pitteway. Algorithm for drawing ellipses or hyperbolae with digital plotter. *Computer Journal*, 10:282, 1967-1968. 1
4. B.W. Jordan, W.J. Lennon, and B.D Holm. An Improved Algorithm for the Generation of Nonparametric Curves. *IEEE TOC*, C-22(12), 1973. 1
5. J.-J. Bourdin and J.-P. Braquelaire. Color Shading in 2D Synthesis. In *Proceedings of Eurographics'90*, pages 41–49,547–548, 1990. 3


```
#define FUNCTION a*x*x-y
#define DERIVATEX +2*a*x
#define DERIVATEY -1
```

Figure 2: *Results for* $f_r(x, y) = ax^2 - y$.

```
#define FUNCTION a*x+b*y+c
#define DERIVATEX +a*x
#define DERIVATEY +b*y
```

Figure 3: *Results for* $f_r(x, y) = a * x + b * y + c$.

```
#define FUNCTION x*x+y*y-r*r
#define DERIVATEX +2*x
#define DERIVATEY +2*y
```

Figure 4: *Results for* $f_r(x, y) = x * x + y * y - r * r$.