Curve Fitting and Animated Drawing Thai Painting

Nunnapad Toadithep[1] and Paramat Itisarn[2]

nisachol@cp.su.ac.th
[1,2]Faculty of Science, Silpakorn University, Nakorn Pathom 73000
Thailand

Abstract : *My work is to develop the software to draw Thai Painting, Kanok is the accient and at high level of line drawing. Kanok usually use in the Thai Lord way of life from the past time to now, decorate the property in celebration especially for the Royal. Young Thai people only know this art but have no a ability of drawing this arts. My work aims to make young Thai people familiar and appreciate this arts. Tool of developing this work are image processing and animation in VPython with Bézier curve fitting. This software mimic the line pattern of Kanok and draw in animation way, showing every step of drawing. This software, an Artifitial Artist draw every graphics of Thai Paintings well-done as an artist.*

1. Introduction

Thai painting or Thai line pattern is the part of Thai Fine Arts, is very delicate of drawing. There are not so many persons who specialize in this arts. People of one country should be proud of their national arts. The national arts is the best thing to express the whole people. It time to make a tool for supporting and spreading the ability of this arts to all of young Thai people. This software not only for Thai people but also for all in other country to know and have a sight with Thai Fine Arts.

Drawing is to make the connect of points and follow the line pattern of Arts. The drawn line can not a beautiful line without the technique of curve fitting, so that the Bézier curve fitting is chosen to apply to my work. And the second helpful tool of this work is VPython, which have a tool to draw a curve and can display in some delay time for providing the user of animation viewing.

2. Curves and Thai Line Pattern

2.1 Bézier Curve

Bézier curves were widely publicized in 1962 by the French engineer Pierre Bézier, who used them to design automobile bodies. The curves were first developed in 1959 by Paul de Casteljau using de Casteljau's algorithm, a numerically stable method to evaluate Bézier curves.

2.1.1 Linear Bézier curves[1]

Given points $\mathbf{P}_0$ and $\mathbf{P}_1$, a linear Bézier curve is simply a straight line between those two points. The curve is given by (equation 2.1)

$$B(t) = P_0 + t\,(P_1 - P_0) = (1 - t)\,P_0 + tP_0 \quad , t \in [0,1] \tag{2.1}$$

and is equivalent to linear interpolation.

2.1.2 Quadratic Bézier curves[1]

A quadratic Bézier curve is the path traced by the function **B**(*t*), given points $\mathbf{P}_0$, $\mathbf{P}_1$, and $\mathbf{P}_2$, **as** equation 2.2 and depict as graph in Figure 2.1.

$$B(t) = (1-t)^2 P_0 + 2t(1-t) P_1 + t^2 P_2 \quad ,t \in [0,1] \tag{2.2}$$

A quadratic Bézier curve is also a parabolic segment. TrueType fonts use Bézier splines composed of quadratic Bézier curves.

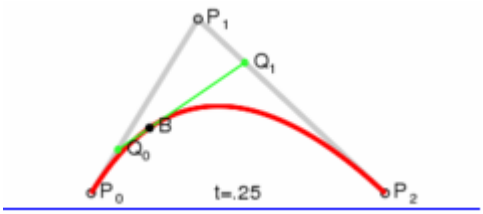

Figure 2.1 Construction of a quadratic Bézier curve [1]

2.1.3 Cubic Bézier curves[1]

Four points $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$ and $\mathbf{P}_3$ in the plane or in three-dimensional space define a cubic Bézier curve. The curve starts at $\mathbf{P}_0$ going toward $\mathbf{P}_1$ and arrives at $\mathbf{P}_3$ coming from the direction of $\mathbf{P}_2$. Usually, it will not pass through $\mathbf{P}_1$ or $\mathbf{P}_2$; these points are only there to provide directional information. The distance between $\mathbf{P}_0$ and $\mathbf{P}_1$ determines "how long" the curve moves into direction $\mathbf{P}_2$ before turning towards $\mathbf{P}_3$. The parametric form of the curve is:

$$B(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3 \quad , t \in [0,1] \tag{2.3}$$

Modern imaging systems like PostScript, Asymptote and Metafont use Bézier splines composed of cubic Bézier curves for drawing curved shapes. Figure 2.2 show examples of cubic B´ezier curves.

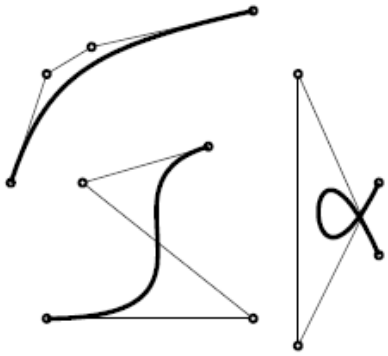

Figure 2.2 Examples of cubic B´ezier curves [2]

2.1.4 Generalization[1]

The Bézier curve of degree *n* can be generalized as follows. Given points $\mathbf{P}_0$, $\mathbf{P}_1$,..., $\mathbf{P}_n$, the Bézier curve is

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i$$
$$= (1-t)^n P_0 + \binom{n}{1}(1-t)^{n-1} tP_1 + \ldots + t^n P_n \qquad , t \in [0,1] \tag{2.4}$$

for example, for $n = 5$:

$$B(t) = (1-t)^5 P_0 + 5t(1-t)^4 P_1 + 10t^2(1-t)^3 P_2 + 10t^3(1-t)^2 P_3 + 5t^4(1-t)P_4 + t^5 P_5 \quad , t \in [0,1] \tag{2.5}$$

The Linear, Quadratic , and Cubic Bézier curves can be said as B´ezier curves in degree 1, degree 2 , and degree 3 respectively. And the higher degree such as degree 4 is composed of 5 points as shown in Figure 2.3.

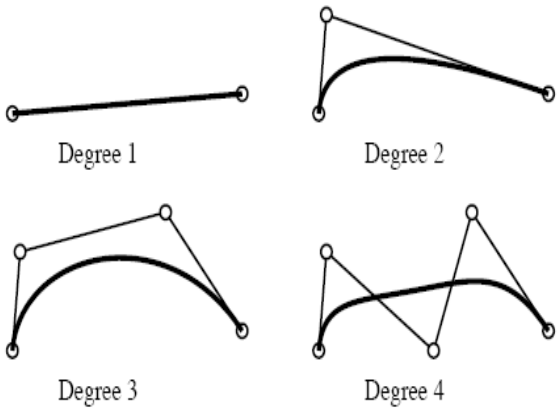

Figure 2.3 B´ezier curves of various degree [2]

2.1.5 Terminology[1]

Some terminology is associated with these parametric curves. We have

$$B(t) = \sum_{i=0}^{n} b_{i,n}(t) P_i \qquad , t \in [0,1] \tag{2.6}$$

where the polynomials

$$b_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} \qquad , i=0,\ldots n \tag{2.7}$$

are known as Bernstein basis polynomials of degree *n*, defining $t^0 = 1$ and $(1-t)^0 = 1$.

The points $\mathbf{P}_i$ are called *control points* for the Bézier curve. The polygon formed by connecting the Bézier points with lines, starting with $\mathbf{P}_0$ and finishing with $\mathbf{P}_n$, is called the *Bézier polygon* (or *control polygon*). The convex hull of the Bézier polygon contains the Bézier curve.

2.1.6 Example [3]

Find the Bézier curve which starts at $\{x_0, y_0\} = \{2, 2\}$ and ends at $\{x_3, y_3\} = \{4, 1\}$ which has the control points $\{x_1, y_1\} = \{0, 1\}$ and $\{x_2, y_2\} = \{3, -1\}$, respectively. Use Bernstein polynomials.

Result of this problem, in parametric equation of points of curve with t interval 0,1 is

$$\{ \quad 2(1 - t)^3 + 9(1 - t)t^2 + 4t^3 \ , \\ 2(1 - t)^3 + 3(1 - t)^2 t - 3(1-t)t^2 + t^3 \ \} \qquad (2.8)$$

mean that

$$x = 2(1 - t)^3 + 9(1 - t)t^2 + 4t^3 \quad \text{and} \\ y = 2(1 - t)^3 + 3(1 - t)^2 t - 3(1-t)t^2 + t^3 \qquad (2.9)$$

for $t \in [0,1]$

2.2 Kanok

Kanok is one type of Thai Line Pattern, and Kanok itself have been divided in many type of Kanok , as shown in Figure 2.4. In Figure 2.5 depict figure of Thai Fine Arts which apply the painting of Kanok line patterns.

Figure 2.4 The image of some type of Kanok

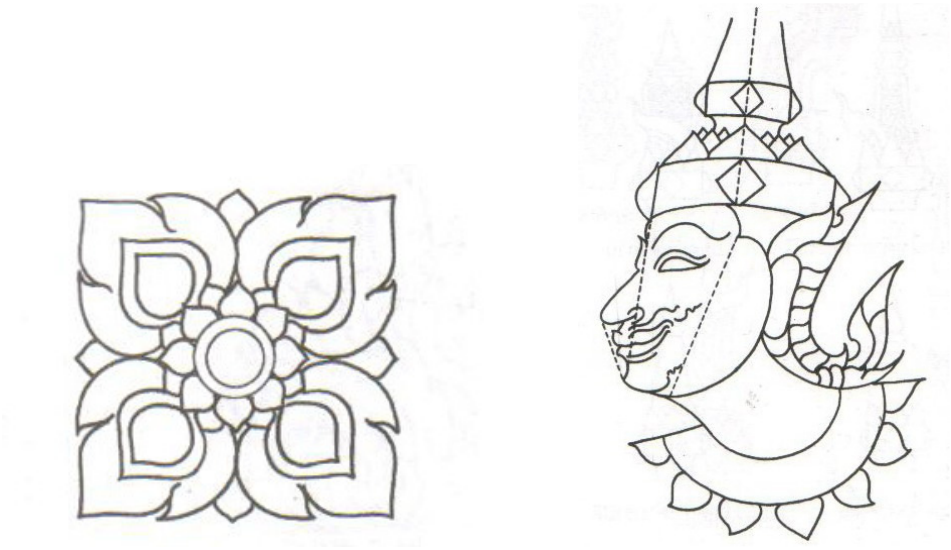

Figure 2.5 The image of Thai Fine Arts, applying Kanok painting

3. VPython [9]

VPython is the Python programming language plus a 3D graphics module called "Visual" developed by David Scherer. This document describes all of the Visual capabilities. To invoke the Visual module, place the following statement at the start of the file:

```
from visual import *
```

There are Basic Display Objects in VPython such as cylinder, arrow, sphere etc. and curve which used in drawing line pattern.

3.1 The curve Object[9]

The curve object displays straight lines between points, and if the points are sufficiently close together you get the appearance of a smooth curve. In addition to its basic use for displaying curves, the curve object has powerful capabilities for other uses, such as efficient plotting of functions. Example of make curve using Curve Object in VPython.

```
curve (pos=[(0,0,0), (1,0,0), (2,1,0)], radius=0.05)
```

3.2 Adding more points to a curve[9]

Curves can be created incrementally with the append() function. A new point by default shares the characteristics of the last point. Example : create the helix cure, append the point (sin (2*t), cos(2*t), t) , t from 0 to 4*pi can be written in VPython like this, and the result shown in figure 3.1.

```
helix = curve(color=color.yellow)
for t in arange(0, 4*pi, 0.1):
    helix.append ( pos=(sin(2*t),cos(2*t), t) )
```

Note that The curve should be named.

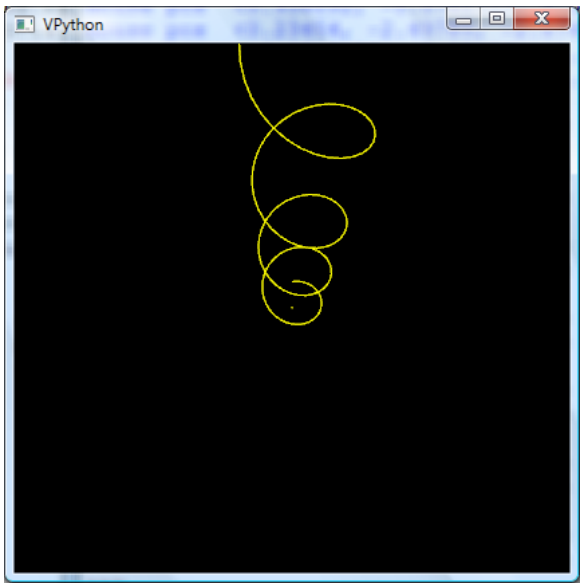

Figure 3.1 Curve forming by appending the point

3.3 Animation with VPython [9]

Everyone know that the computer can execute many million of statements in only one second. So that we can not see the object in display window running, say that the ball go right. Visual make you: can create an animation (can see what is going on) with rate command.

rate(frequency)

this command is to limit the Animation Rate, halts computations until 1.0/frequency seconds after the previous call to rate().

For example, rate(50) will halt computations long enough to make sure that at least 1.0/50.0 second has elapsed (if this much time has already elapsed, no halt is performed). If you place rate(50) inside a computational loop, the loop will execute at a maximum of 50 times per second, even if the computer can run faster than this. This makes animations look about the same on computers of different speeds, as long as the computers are capable of carrying out 50 computations per second. For example the sphere named: ball change its position in x go to the right each time 0.5 unit. If without command rate(50), the user can not see the ball go right.

```
from visual import *
ball = sphere (radius=.2, color= color.orange)
while  1:
    rate (50)
    ball.x = ball.x + 0.5
```

Note:
ball.x = ball.x +.5 be replaced with ball.pos = ball.pos + (.5,*0,0)*

4. Work steps

4.1 Step of work in process

1.Studying of function for curve fitting
2.Studying the drawing of Kanok line pattern
3.Studying the Kanok
4.Studying curve drawing and animation in VPython
5.Getting the prototype image, from scanning or other input
6.Writing program accept the points which overlay on the prototype image for forming the Kanok line pattern
7.Writing program to display the set of points to smooth curve using Bezier curve fitting
8.Use the set of points of step 7, as a input of program, displaying the Kanok line pattern

Note :

if the image of Kanok is a color image or have not only lines. The image should be converted to grayscale images and apply edge detection to the image.

4.2 Some functions in mywork

Here is the code to accept the point from detecting the mouse click and fit the curve with Bézier curve fitting method, where L is list of points of curve. Figure 3.2 show the curve with Bézier curve fitting method.

```
while 1:
    m = scene.mouse.getevent()
    if m.click == "right" :        #exit loop
        break
    elif   m.click == 'left':
          sphere(pos = m.pos, radius=.05)
          L.append(m.pos)
          c.append(m.pos)

DrawBezier(L)   # Drawing the curve
```

Here are the related functions fact(n), Comb(n, r), and DrawBezier(L)

```
def fact(n):
  if n==0:
    return 1
  else:
    return n*(fact(n-1))
```

```
def Comb(n,r):
    return (fact(n)/ (fact(n-r)*fact(r)))
def  DrawBezier(L):
    n=len(L)-1
    c1=curve(color=color.yellow)
    c1.append(L[0])
    for  t  in  arrange (0, 1, .05):
            rate(50)
            x,y=0.0, 0.0
            for i in range(0,n+1,1):
                a=Comb(n,i)
                x=x+ L[i].x*a*(t**i) * (1-t)**(n-i)    // x generated by Bezier curve fitting
                y=y+ L[i].y*a*(t**i) * (1-t)**(n-i)    // y generated by Bezier curve fitting
            c1.append( (x,y) )
     return c1
```

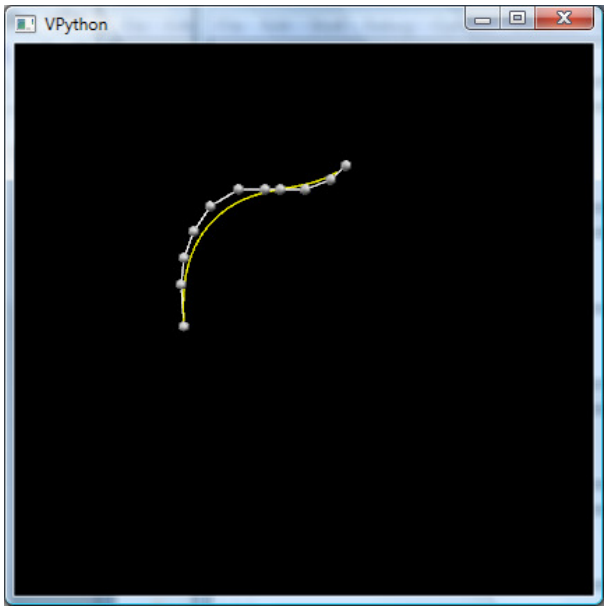

Figure 3.2 Curve in VPython after applying the cubic B´ezier curve fitting

4.3 Result of work

The program can learn the line pattern of Kanok by user clicking on the image and save the set of points and using Bézier curve fitting method to smooth the curve**,** and then draw automatically in animation way. Figure 4.1, Figure 4.2, and Figure 4.3 are the results of program.

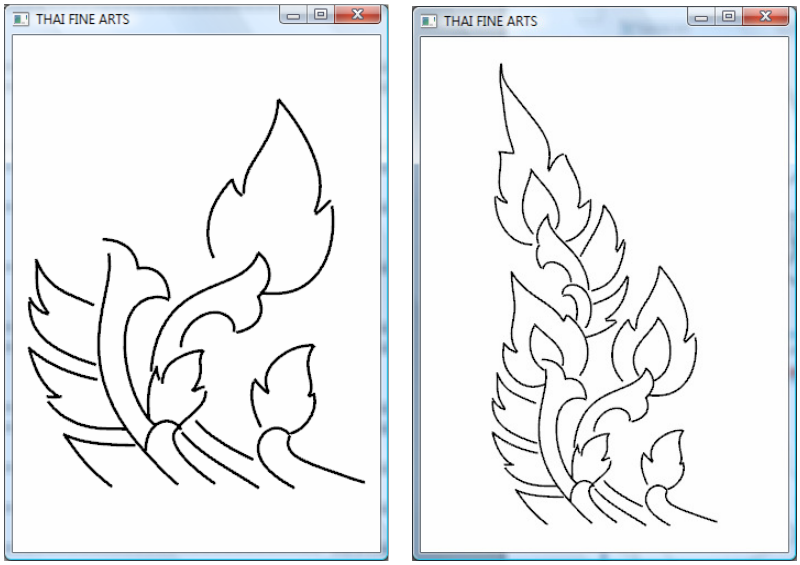

Figure 4.1 The animation of drawing Kanok

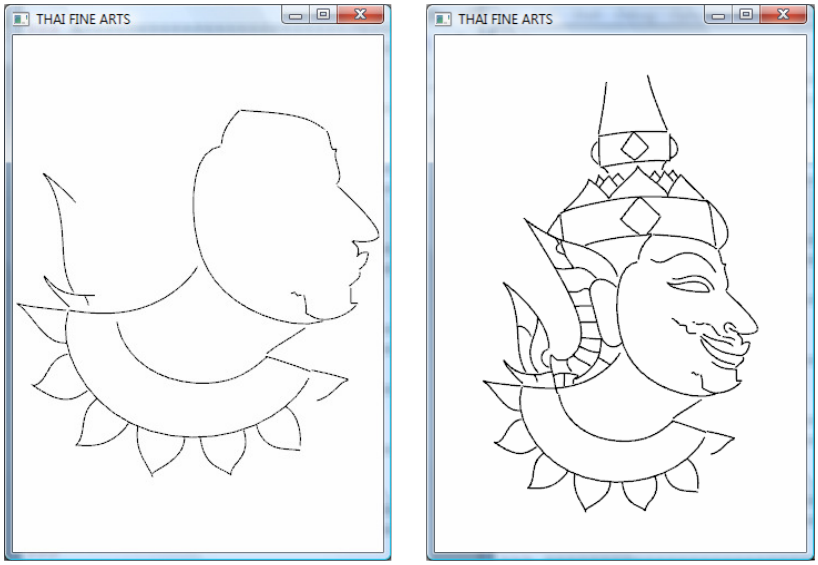

Figure 4.2 The animation of drawing Thai line pattern

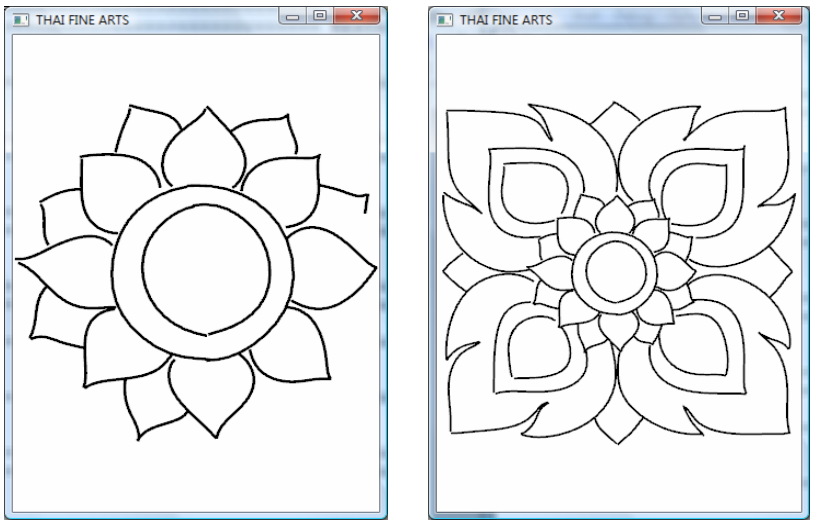

Figure 4.3 The animation of drawing Thai line pattern

5. Conclusion

This work is completed and accomplished by the Objectives. This program can learn the pattern of Kanok with some entries plot of curve from user or collector. Program create set of points using Bezier curve fitting technique, which is written in VPython. It is a useful work if have been applied in teaching the young Thai people for appreciation their national culture and realizing the history of Thai Accient Arts.

References

[1] http://en.wikipedia.org/wiki/Bezier_curves
[2] http://www.tsplines.com/resources/class_notes/Bezier_curves.pdf
[3] http://math.fullerton.edu/mathews/n2003/Bezier CurveMod.html
[4] http://www.pages.drexel.edu/~weg22/can_tut.html
[5] http://en.wikipedia.org/wiki/Edge_detection
[6] http://en.wikipedia.org/wiki/Grayscale
[7] http://www.codersource.net/csharp_color_image_to_binary.aspx
[8] http://www.archive.org/details/Lectures_on_Image_Processing
[9] http://www.vpython.org