Computer Graphics Assignment 1

* Name:Yongxi Huang Student ID:119033910011 Email: huangyongxi@sjtu.edu.cn

Code will be uploaded to
`https://github.com/Riften/SJTU-Computer-Graphics-2020-Assignments`
after deadline.

Design an incremental algorithm for the given polynomial:

$$y = ax^3 + bx^2 + cx + d(1 \le x \le 100)$$

(without multiplication)

1 Algorithm design

Here we define

$$\begin{aligned} x_{i+1} &= x_i + 1 \\ y_i &= ax_i^3 + bx_i^2 + cx_i + d \\ dy_i &= y_{i+1} - y_i \\ ddy_i &= dy_{i+1} - dy_i,\ (i \in \mathbb{Z}) \end{aligned}$$

When the value of x increases by 1, there is

$$\begin{aligned} y_{i+1} &= a(x_i+1)^3 + b(x_i+1)^2 + c(x_i+1) + d \\ &= (ax_i^3 + bx_i^2 + cx_i + d) + 3ax_i^2 + (3a+2b)x_i + a + b + c \\ &= y_i + 3ax_i^2 + (3a+2b)x_i + a + b + c \\ &= y_i + dy_i \qquad (1.1) \\ dy_i &= 3ax_i^2 + (3a+2b)x_i + a + b + c \end{aligned}$$

dy_i can be derived from dy_{i-1} by replacing x_i with $(x_{i-1}+1)$.

$$\begin{aligned} dy_i &= 3a(x_{i-1}+1)^2 + (3a+2b)(x_{i-1}+1) + a + b + c \\ &= 3ax_{i-1}^2 + (3a+2b)x_{i-1} + a + b + c + 6ax_{i-1} + 6a + 2b \\ &= dy_{i-1} + 6ax_{i-1} + 6a + 2b \\ &= dy_{i-1} + ddy_{i-1} \qquad (1.2) \\ ddy_{i-1} &= 6ax_{i-1} + 6a + 2b \end{aligned}$$

We further derive ddy_{i-1} in the same way.

$$\begin{aligned} ddy_{i-1} &= 6a(x_{i-2}+1) + 6a + 2b \\ &= ddy_{i-2} + 6a \qquad (1.3) \end{aligned}$$

According to equation (1.1), (1.2), and (1.3), the incremental function is

$$\begin{aligned} y_{i+1} &= y_i + dy_{i-1} + ddy_{i-2} + 6a \\ &= y_i + (y_i - y_{i-1}) + (dy_{i-1} - dy_{i-2}) + 6a \\ &= 3y_i - 3y_{i-1} + y_{i-2} + 6a \qquad (1.4) \end{aligned}$$

Based on incremental function (1.4), the incremental drawing algorithm for cubic curve is shown in Alg.1.

Algorithm 1: Incremental Drawing Algorithm for Cubic Curve.

```
   Input: $a, b, c, d, x_0, x_n$
   Result: Draw the line $y = ax^3 + bx^2 + cx + d$ from $x_0$ to $x_n$
 1 Compute $\{y_0, y_1, y_2\}$ by $y_i = a(x_i)^3 + b(x_i)^2 + c(x_i) + d$, $x_i = x_0 + i$
 2 DrawPixels($x_0, y_0, x_1, y_1$)
 3 DrawPixels($x_1, y_1, x_2, y_2$)
 4 for $i = 3; i \leq n; i++$ do
 5     $x_i = x_0 + i$
 6     $y_i = 3y_{i-1} - 3y_{i-2} + y_{i-3} + 6a$
 7     DrawPixels($x_{i-1}, y_{i-1}, x_i, y_i$)
```

2 Implementation

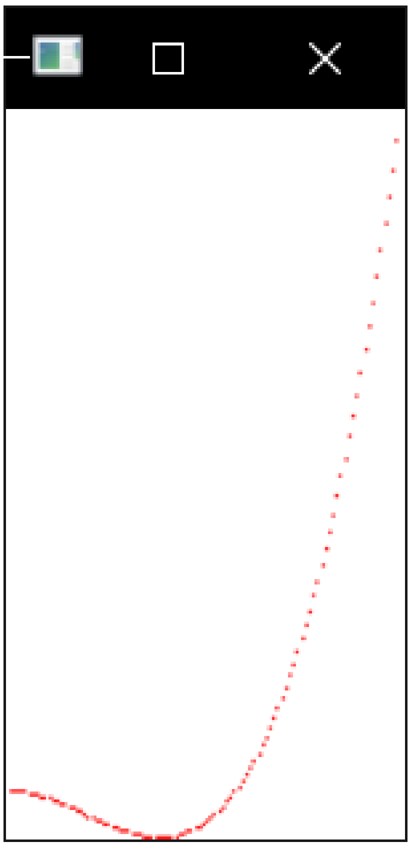
The program is implemented in **c++** with **OpenGL**. It uses function $glVertex2i()$ to draw the curve pixel-by-pixel incrementally. In order to show most of the curve, the view window would be moved to the position of curve.

A vital part of code is the implementation of $DrawPixels(x1, y1, x2, y2)$ in Alg.1. Note that we should not simply draw two points. Otherwise the curve would be intermittent as shown in Fig.1(a). Bresenham's Algorithm is an ideal solution to this problem. However, in this implementation x only grows 1 at each step. So $DrawPixels(x1, y1, x2, y2)$ is designed as shown in Alg.2 which is more simple but get the same result as Bresenham's Algorithm.

The drawing result for curve $y = 0.0005x^3 - 0.03x^2 + 0.05x + 100$ is shown in Fig.2.1.

Algorithm 2: DrawPixels(x_0, y_0, x_1, y_1)

```
   Input: $\{x_0, y_0, x_1, y_1\}$ where $x_1 = x_0 + 1$.
   Result: Draw the pixels between $(x_0, y_0)$ and $(x_1, y_1)$.
 1 $y_0 \leftarrow Round(y_0)$
 2 $y_1 \leftarrow Round(y_1)$
 3 if $y_0 == y_1$ then
 4     WritePixel($x_0, y_0$);
 5 else
 6     $mid \leftarrow (y_0 + y_1)/2$ ;Can be implemented by right shift 1 bit.
 7     for $y \leftarrow y_0$ to $mid$ do
 8         WritePixel($x_0, y$)
 9     for $y \leftarrow mid$ to $y_1$ do
10         WritePixel($x_1, y$)
```


(a) Naive method

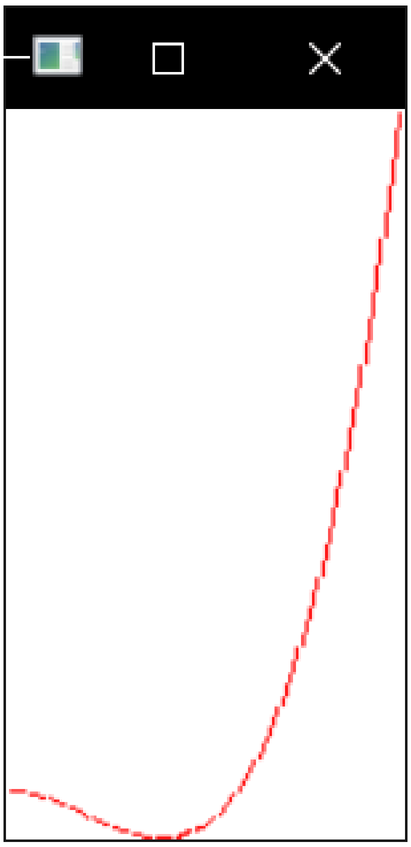

(b) Optimized

Figure 2.1: Result for $y = 0.0005x^3 - 0.03x^2 + 0.05x + 100$