

作业 1

Log Creative

2024 年 3 月 14 日

目 录

1 问题	2
2 递增式算法	2
2.1 算法描述	2
2.2 伪代码	2
2.3 代码实现	3
3 命令程序	5
3.1 运行方式	5
3.2 代码实现	5
4 OpenGL 展示	6
4.1 运行方式	6
4.2 代码实现	6
参考文献	9
A 可执行文件	9
A.1 Windows	9
A.2 macOS	10
A.3 Linux	10
B 源代码编译	10
B.1 Windows	10
B.2 macOS	10
B.3 Linux	10

1 问题

Design an incremental algorithm for the given polynomial:

$$y = ax^2 + bx + c \quad (x_b \leq x \leq x_e) \quad (1)$$

(without any multiplication)

2 递增式算法

2.1 算法描述

对于多项式 (1), 可以得到横坐标相差 1 时纵坐标的差分值

$$\Delta y(x) = y(x+1) - y(x) = 2ax + a + b \quad (2)$$

当知道起始点 x_b 的纵坐标 $y(x_b)$ 时, 就可以通过差分值得到区间内任意一点的纵坐标

$$y(x) = y(x_b) + \sum_{i=0}^{x-x_b-1} \Delta y(x_b + i) \quad (3)$$

但是每次都计算差分值实际上会有一定的重复计算 (当认为乘法计算代价高的时候), 所以考虑使用二阶差分来进一步加速运算

$$\Delta[\Delta y(x)] = 2ax + a + b - [2a(x-1) + a + b] = 2a = a + a \quad (4)$$

那么式 (3) 就变成

$$y(x) = y(x_b) + \sum_{i=0}^{x-x_b-1} [\Delta y(x_b) + (a+a)i] \quad (5)$$

或者写成迭代的形式

$$y(x+1) = y(x) + \Delta y(x) = y(x) + [\Delta y(x-1) + (a+a)] \quad (6)$$

就可以实现全加法的迭代运算。式 (5) 提示我们需要预先知道 $y(x_b)$ 和 $\Delta y(x_b)$, 本文将会从 0 出发, 使用累加的方式实现乘法, 算出这两个初值, 具体实现见第 2.3 节的 `getValue(x)` 函数和 `polyItem(coeff, base, order)` 函数。

2.2 伪代码

主要伪代码见算法 1。

算法 1 递增式计算抛物线

Input: 多项式系数 a, b, c , 起始点 x_b , 终止点 x_e

Output: 区间 $[x_b, x_e]$ 对应的多项式结果 y

```

1 使用累加计算  $y(x_b) = ax_b^2 + bx_b + c$ ,  $\Delta y(x_b) = 2ax_b + a_b$ ;
2 设定变量初值  $y \leftarrow y(x_b)$ ,  $\Delta y \leftarrow \Delta y(x_b)$ ;
3 foreach  $x \leftarrow x_b$  to  $x_e$  do
4    $y(x) \leftarrow \text{round}(y)$ ;
5    $y \leftarrow y + \Delta y$ ;
6    $\Delta y \leftarrow \Delta y + a + a$ ;
7 end
8 return  $y$ ;
```

2.3 代码实现

代码实现于 `IncrPoly` 类中, 算法 1 主要实现于 `getRangeValue()` 函数, 它会调用 `getValue(x)` 来计算初值, 这两个函数都会调用 `polyItem(coeff, base, order)` 函数来计算单个多项式项的值。

其中 `polyItem(a, x, r)` 采用累加、迭代的形式实现对多项式项 ax^r 的计算: 对于高阶项 ($r > 1$) 会逐步化归为 1 阶的情况, 对于 1 阶的情况, 采用从 0 的位置开始逐步累加 (减) 浮点数的形式得到对应的值。

$$ax^r = \text{polyItem}(a, x, r) = \begin{cases} a, & r = 0; \\ \underbrace{a + \dots + a}_{\#a=x}, & r = 1 \text{ and } x \geq 0; \\ \underbrace{-a - \dots - a}_{\#a=x}, & r = 1 \text{ and } x < 0; \\ \text{polyItem}(ax^{r-1}, x, 1), & r > 1. \end{cases} \quad (7)$$

Listing 1 `../source/IncrPoly/IncrPoly.hpp`

```

1 #ifndef INCRPOLY_INCRPOLY_HPP
2 #define INCRPOLY_INCRPOLY_HPP
3
4 class IncrPoly {
5     float a;
6     float b;
7     float c;
8
9     static int floor(float y) { return y >= 0.0f ? (int) y : (int) y - 1; }
10
11     static int round(float y) { return floor(y + 0.5f); }
12 }
```



```
13 public:
14     IncrPoly(float a, float b, float c) : a(a), b(b), c(c) {}
15
16     // Calculate coeff*baseorder.
17     float polyItem(float coeff, int base, int order) {
18         float value = 0;
19         if (order == 0) return coeff;
20         if (order == 1) {
21             if (base >= 0) {
22                 for (int i = 0; i < base; ++i) value += coeff;
23             } else {
24                 for (int i = 0; i < -base; ++i) value -= coeff;
25             }
26             return value;
27         }
28         return polyItem(polyItem(coeff, base, order - 1), base, 1);
29     }
30
31     // Calculate  $ax^2+bx+c$  for the initial value.
32     float getValue(int x) {
33         float value = c;
34         value += polyItem(b, x, 1);
35         value += polyItem(a, x, 2);
36         return value;
37     }
38
39     // Calculate  $ax^2+bx+c$  for the range  $[x_b, x_e]$ ,
40     // store the result into the array  $*y\_output = \text{int}[x_e - x_b + 1]$ .
41     void getRangeValue(int x_b, int x_e, int *y_output) {
42         if (x_b > x_e) {
43             // swap the value when the left is bigger than the right.
44             int x_tmp = x_b;
45             x_b = x_e;
46             x_e = x_tmp;
47         }
48         int x = x_b;
49         float y = getValue(x);
50         const float two_a = a + a;
51         //  $a(x+1)^2 + b(x+1) + c - ax^2 - bx - c = 2ax + a + b$ 
52         float incr = polyItem(two_a, x, 1) + a + b;
53         for (; x <= x_e; ++x) {
54             *y_output++ = round(y);
55             y += incr;
56             //  $2a(x+1) + a + b - 2ax - a - b = 2a$ 
57             incr += two_a;
```

```

58     }
59 }
60
61 virtual ~IncrPoly() = default;
62 };
63
64
65 #endif // INCRPOLY_INCRPOLY_HPP

```

3 命令行程序

3.1 运行方式

```

Incremental algorithm for calculating  $a \cdot x^2 + b \cdot x + c$  within range  $[x_b, x_e]$ .
Please input the coefficients (could be decimals); split by whitespace and hit Enter [a b c]: -0.05 5 10
Please input the range boundaries (make sure they are integers); split by whitespace and hit Enter [x_b x_e]: -30 130
The polynomial result are (rounded, from the left boundary to the right boundary): -185 -177 -169 -161 -154 -146 -139 -131 -124 -117 -110 -103 -96 -89 -83 -76

```

图 1 命令行程序输出结果

运行命令行程序，如图 1 所示。首先输入三个参数（用空格分隔），比如 -0.05 5 10，按下回车；接着输入左右边界（用空格分隔，如果输入不是整数，将会转换为整数），比如 -30 130，按下回车；最后就会输出这个区间之间纵坐标的整数结果。

3.2 代码实现

命令行程序的代码展示了如何基本地调用 IncrPoly 类。

Listing 2 ../source/IncrPoly/main.cpp

```

1 #include "IncrPoly-cli.h"
2 #include "IncrPoly.hpp"
3
4 int main() {
5     float a, b, c;
6     int x_b, x_e;
7     prepare_input(a, b, c, x_b, x_e);
8     IncrPoly poly(a, b, c);
9     int length = x_e - x_b + 1;
10    int *y = new int[length];
11    poly.getRangeValue(x_b, x_e, y);
12    output_result(y, length);
13    delete[] y;
14    return 0;
15 }

```

其中 IncrPoly-cli.h 主要实现了终端输入提示逻辑。

Listing 3 `../source/IncrPoly/IncrPoly-cli.h`

```
1  #ifndef INCRPOLY_OPENGL_INCRPOLY_CLI_H
2  #define INCRPOLY_OPENGL_INCRPOLY_CLI_H
3
4  #include <iostream>
5
6  void prepare_input(float &a, float &b, float &c, int &x_b, int &x_e) {
7      std::cout << "Incremental algorithm for calculating  $a*x^2+b*x+c$  "
8          "within range [x_b,x_e]." << std::endl;
9      std::cout << "Please input the coefficients (could be decimals); "
10         "split by whitespace and hit Enter [a b c]: ";
11      std::cin >> a >> b >> c;
12
13      float _x_b, _x_e;
14      std::cout << "Please input the range boundaries (make sure they are integers); "
15         "split by whitespace and hit Enter [x_b x_e]: ";
16      std::cin >> _x_b >> _x_e;
17      if (_x_b > _x_e) {
18          float _x_tmp = _x_b;
19          _x_b = _x_e;
20          _x_e = _x_tmp;
21      }
22      x_b = (int) _x_b;
23      x_e = (int) _x_e;
24  }
25
26  void output_result(int *y, int length) {
27      std::cout << "The polynomial result are (rounded, "
28         "from the left boundary "
29         "to the right boundary): ";
30      for (int i = 0; i < length; ++i) {
31          std::cout << *y++ << ' ';
32      }
33      std::cout << std::endl;
34  }
35
36  #endif //INCRPOLY_OPENGL_INCRPOLY_CLI_H
```

4 OpenGL 展示

4.1 运行方式

运行图形程序，按照第 3.1 节的终端输入方法输入相关数值，然后就可以在弹出的窗口中看到如图 2 所示的可视化结果。

4.2 代码实现

基本框架对照 LearnOpenGL 《你好，三角形》一节^[1] 设置，其中头文件 shader_s.h 中的着色器类为 LearnOpenGL 《着色器》一节^[2] 的源代码^[3]，会加载顶点着色器 vertexShader.glsl 和片段着色器 fragmentShader.glsl。

Listing 4 `../source/src/main.cpp`

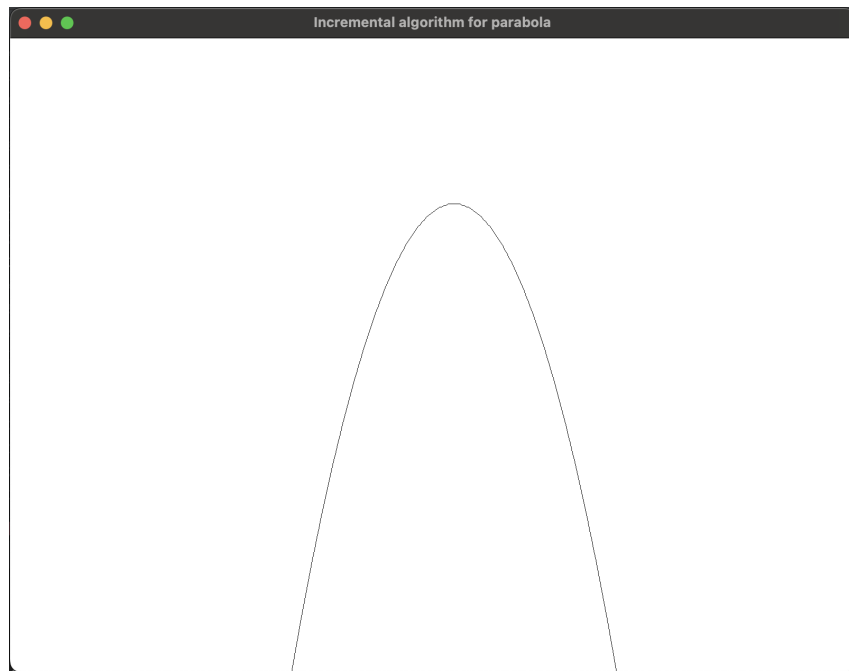


图2 图形程序输出结果, 输入 $a = -0.05, b = 5, c = 10, x_b = -30, x_e = 130$

```
1  #include "glad/glad.h"
2  #include "GLFW/glfw3.h"
3  #include "shader_s.h"
4  #include "../IncrPoly/IncrPoly-cli.h"
5  #include "../IncrPoly/IncrPoly.hpp"
6
7  int main()
8  {
9      float a, b, c;
10     int x_b, x_e;
11     prepare_input(a, b, c, x_b, x_e);
12     IncrPoly poly(a, b, c);
13     int length = x_e - x_b + 1;
14     int* y = new int[length];
15     poly.getRangeValue(x_b, x_e, y);
16
17     /* Initialize the library */
18     if (!glfwInit()) {
19         return -1;
20     }
21
22     /* Create a windowed mode window and its OpenGL context */
23 #ifdef __APPLE__
24     /* We need to explicitly ask for a 3.3 context on Mac */
25     glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
26     glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
27     glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
28     glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
29 #endif
30     GLFWwindow* window = glfwCreateWindow(800, 600, "Incremental algorithm for parabola", nullptr, nullptr);
31     if (!window) {
32         glfwTerminate();
```



```
33     return -1;
34 }
35
36 /* Make the window's context current */
37 glfwMakeContextCurrent(window);
38
39 /* Initialize glad (loads the OpenGL functions) */
40 if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress)) {
41     return -1;
42 }
43
44 /* Create the App */
45 int w, h;
46 glfwGetWindowSize(window, &w, &h);
47
48 Shader ourShader("assets/shader/vertexShader.glsl", "assets/shader/fragmentShader.glsl");
49 ourShader.use();
50
51 int coordLength = length + length;
52 int* vertices = new int[coordLength];
53 int x = x_b;
54 int* y_ptr = y;
55 int* vertices_ptr = vertices;
56 for (int i = 0; i < length; ++i) {
57     std::cout << "(" << x << " " << *y_ptr << " ) ";
58     *vertices_ptr++ = x++;
59     *y_ptr++ = *y_ptr++;
60 }
61 delete[] y;
62
63 unsigned int VBO;
64 glGenBuffers(1, &VBO);
65
66 unsigned int VAO;
67 glGenVertexArrays(1, &VAO);
68 glBindVertexArray(VAO);
69 glBindBuffer(GL_ARRAY_BUFFER, VBO);
70 glBufferData(GL_ARRAY_BUFFER, sizeof(*vertices) * coordLength, vertices, GL_STATIC_DRAW);
71 glVertexAttribPointer(0, 2, GL_INT, GL_FALSE, 2 * sizeof(int), (void*)0);
72 glEnableVertexAttribArray(0);
73
74 /* Loop until the user closes the window */
75 while (!glfwWindowShouldClose(window)) {
76
77     glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
78     glClear(GL_COLOR_BUFFER_BIT);
79
80     ourShader.use();
81     glBindVertexArray(VAO);
82     glDrawArrays(GL_LINE_STRIP, 0, length);
83
84     /* Swap front and back buffers */
85     glfwSwapBuffers(window);
86
87     /* Poll for and process events */
88     glfwPollEvents();
89 }
90
91 glDeleteVertexArrays(1, &VAO);
92 glDeleteBuffers(1, &VBO);
93 delete[] vertices;
94
```




```
95     glfwTerminate();  
96     return 0;  
97 }
```

Listing 5 `../source/assets/shader/vertexShader.glsl`

```
1 #version 330 core  
2 layout (location = 0) in vec2 aPos;  
3 void main()  
4 {  
5     gl_Position = vec4((aPos.x-40.0)/200.0, (aPos.y-40.0)/200.0, 0.0, 1.0);  
6 }
```

Listing 6 `../source/assets/shader/fragmentShader.glsl`

```
1 #version 330 core  
2 out vec4 FragColor;  
3  
4 void main()  
5 {  
6     FragColor = vec4(0.0f, 0.0f, 0.0f, 1.0f);  
7 }
```

参考文献

- [1] JoeyDeVries. 你好，三角形 [EB/OL]. 2023. <https://learnopengl-cn.github.io/01%20Getting%20started/04%20Hello%20Triangle/>.
- [2] JoeyDeVries. 着色器 [EB/OL]. 2023. <https://learnopengl-cn.github.io/01%20Getting%20started/05%20Shaders/>.
- [3] JoeyDeVries. 着色器类代码 [EB/OL]. 2023. https://learnopengl.com/code_viewer_gh.php?code=includes/learnopengl/shader_s.h.

A 可执行文件

示例输入：

```
[a b c]: -0.05 5 10  
[x_b x_e]: -100 200
```

注意，需要保证文件夹 `assets/` 存在于运行文件夹中。

A.1 Windows

`IncrPoly-windows.exe` 命令程序，推荐在终端中输入 `../IncrPoly-windows.exe` 以运行查看结果。

`IncrPoly-OpenGL-windows.exe` 图形程序，双击打开。

A.2 macOS

IncrPoly-mac 命令程序。

IncrPoly-OpenGL-mac 图形程序。

A.3 Linux

IncrPoly-linux 命令程序。

IncrPoly-OpenGL-linux 图形程序。

B 源代码编译

本源代码需要使用 CMake^① 编译，推荐使用 CLion 集成环境。

编译方式：

```
1 mkdir cmake-build
2 cd cmake-build
3 cmake ..
```

然后打开相关的工程文件进行编译。

B.1 Windows

必须使用 MSVC 编译（生成 Visual Studio 文件的方法），暂不支持使用 MinGW GCC 编译（会导致运行时动态依赖丢失）。如果使用的是 CLion，需要调整设置：File -> Settings -> Build, Execution and Deployment -> CMake -> Toolchain 选为 Visual Studio。

B.2 macOS

没有特别需要安装的依赖。

B.3 Linux

对于 Ubuntu 需要安装以下依赖：

```
1 sudo apt-get install build-essential
2 sudo apt-get install libx11-dev
3 sudo apt-get install xorg-dev libglu1-mesa-dev
```

^①<https://cmake.org>