上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

作业 1

李子龙
123033910195

2024 年 3 月 14 日

目　录

1 问题

Design an incremental algorithm for the given polynomial:

$$y = ax^2 + bx + c \quad (x_b \le x \le x_e) \tag{1}$$

(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

$$\Delta y(x) = y(x+1) - y(x) = 2ax + a + b \tag{2}$$

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

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

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

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

那么式 (3) 就变成

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

或者写成迭代的形式

$$y(x+1) = y(x) + \Delta y(x) = y(x) + [\Delta y(x-1) + (a+a)] \tag{6}$$

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 $\mathbf{y}$

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

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。

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

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

```
#ifndef INCRPOLY_INCRPOLY_HPP
#define INCRPOLY_INCRPOLY_HPP

class IncrPoly {
   float a;
   float b;
   float c;

   static int floor(float y) { return y >= 0.0f ? (int) y : (int) y - 1; }

   static int round(float y) { return floor(y + 0.5f); }

```

```
public:
    IncrPoly(float a, float b, float c) : a(a), b(b), c(c) {}

    // Calculate coeff*base^order.
    float polyItem(float coeff, int base, int order) {
        float value = 0;
        if (order == 0) return coeff;
        if (order == 1) {
            if (base >= 0) {
                for (int i = 0; i < base; ++i) value += coeff;
            } else {
                for (int i = 0; i < -base; ++i) value -= coeff;
            }
            return value;
        }
        return polyItem(polyItem(coeff, base, order - 1), base, 1);
    }

    // Calculate ax^2+bx+c for the initial value.
    float getValue(int x) {
        float value = c;
        value += polyItem(b, x, 1);
        value += polyItem(a, x, 2);
        return value;
    }

    // Calculate ax^2+bx+c for the range [x_b,x_e],
    // store the result into the array *y_output = int[x_e-x_b+1].
    void getRangeValue(int x_b, int x_e, int *y_output) {
        if (x_b > x_e) {
            // swap the value when the left is bigger than the right.
            int x_tmp = x_b;
            x_b = x_e;
            x_e = x_tmp;
        }
        int x = x_b;
        float y = getValue(x);
        const float two_a = a + a;
        // a(x+1)^2+b(x+1)+c-ax^2-bx-c = 2ax+a+b
        float incr = polyItem(two_a, x, 1) + a + b;
        for (; x <= x_e; ++x) {
            *y_output++ = round(y);
            y += incr;
            // 2a(x+1)+a+b-2ax-a-b = 2a
            incr += two_a;
```

```
        }
    }

    virtual ~IncrPoly() = default;
};


#endif //INCRPOLY_INCRPOLY_HPP
```

3 命令行程序

3.1 运行方式

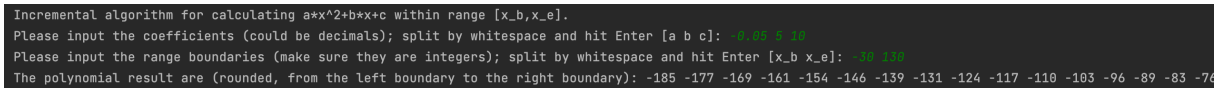

图 1 命令行程序输出结果

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

```
#include "IncrPoly-cli.h"
#include "IncrPoly.hpp"

int main() {
    float a, b, c;
    int x_b, x_e;
    prepare_input(a, b, c, x_b, x_e);
    IncrPoly poly(a, b, c);
    int length = x_e - x_b + 1;
    int *y = new int[length];
    poly.getRangeValue(x_b, x_e, y);
    output_result(y, length);
    delete[] y;
    return 0;
}
```

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

```
#ifndef INCRPOLY_OPENGL_INCRPOLY_CLI_H
#define INCRPOLY_OPENGL_INCRPOLY_CLI_H

#include <iostream>

void prepare_input(float &a, float &b, float &c, int &x_b, int &x_e) {
   std::cout << "Incremental algorithm for calculating a*x^2+b*x+c "
             "within range [x_b,x_e]." << std::endl;
   std::cout << "Please input the coefficients (could be decimals); "
             "split by whitespace and hit Enter [a b c]: ";
   std::cin >> a >> b >> c;

   float _x_b, _x_e;
   std::cout << "Please input the range boundaries (make sure they are integers); "
             "split by whitespace and hit Enter [x_b x_e]: ";
   std::cin >> _x_b >> _x_e;
   if (_x_b > _x_e) {
      float _x_tmp = _x_b;
      _x_b = _x_e;
      _x_e = _x_tmp;
   }
   x_b = (int) _x_b;
   x_e = (int) _x_e;
}

void output_result(int *y, int length) {
   std::cout << "The polynomial result are (rounded, "
             "from the left boundary "
             "to the right boundary): ";
   for (int i = 0; i < length; ++i) {
      std::cout << *y++ << ' ';
   }
   std::cout << std::endl;
}

#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

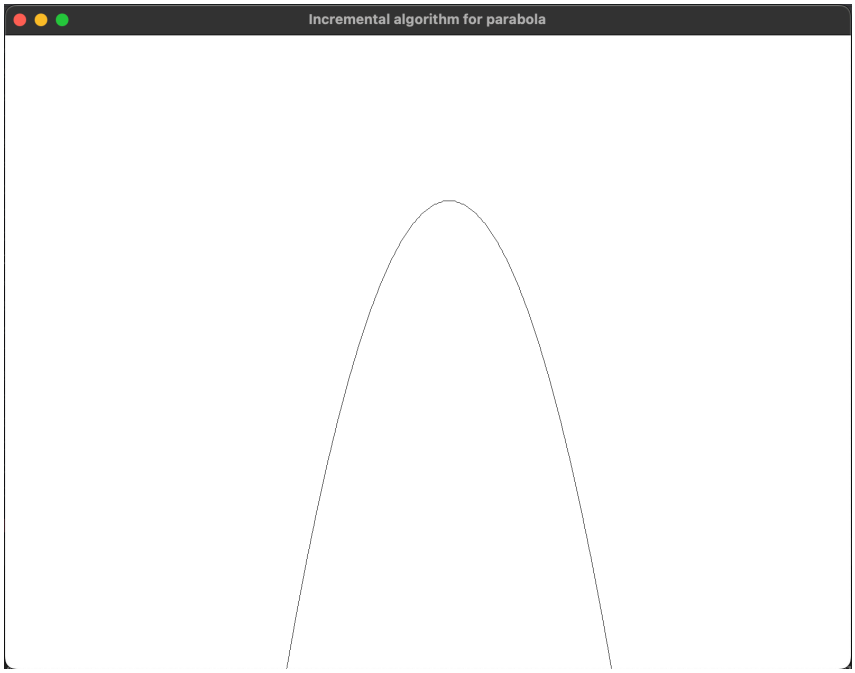

图 2 图形程序输出结果，输入 $a = -0.05, b = 5, c = 10, x_b = -30, x_e = 130$

```cpp
#include "glad/glad.h"
#include "GLFW/glfw3.h"
#include "shader_s.h"
#include "../IncrPoly/IncrPoly-cli.h"
#include "../IncrPoly/IncrPoly.hpp"

int main()
{
   float a, b, c;
   int x_b, x_e;
   prepare_input(a, b, c, x_b, x_e);
   IncrPoly poly(a, b,c);
   int length = x_e - x_b + 1;
   int* y = new int[length];
   poly.getRangeValue(x_b, x_e, y);

   /* Initialize the library */
   if (!glfwInit()) {
      return -1;
   }

   /* Create a windowed mode window and its OpenGL context */
#ifdef __APPLE__
   /* We need to explicitly ask for a 3.3 context on Mac */
   glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
   glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
   glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GL_TRUE);
   glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
#endif
   GLFWwindow* window = glfwCreateWindow(800, 600, "Incremental algorithm for parabola", nullptr, nullptr);
   if (!window) {
      glfwTerminate();
```

```
    return -1;
}

/* Make the window's context current */
glfwMakeContextCurrent(window);

/* Intialize glad (loads the OpenGL functions) */
if (!gladLoadGLLoader((GLADloadproc)glfwGetProcAddress)) {
    return -1;
}

/* Create the App */
int w, h;
glfwGetWindowSize(window, &w, &h);

Shader ourShader("assets/shader/vertexShader.glsl", "assets/shader/fragmentShader.glsl");
ourShader.use();

int coordLength = length + length;
int* vertices = new int[coordLength];
int x = x_b;
int* y_ptr = y;
int* vertices_ptr = vertices;
for (int i = 0; i < length; ++i) {
    std::cout << "(" << x << " " << *y_ptr << ") ";
    *vertices_ptr++ = x++;
    *vertices_ptr++ = *y_ptr++;
}
delete[] y;

unsigned int VBO;
glGenBuffers(1, &VBO);

unsigned int VAO;
glGenVertexArrays(1, &VAO);
glBindVertexArray(VAO);
glBindBuffer(GL_ARRAY_BUFFER, VBO);
glBufferData(GL_ARRAY_BUFFER, sizeof(*vertices) * coordLength, vertices, GL_STATIC_DRAW);
glVertexAttribPointer(0, 2, GL_INT, GL_FALSE, 2 * sizeof(int), (void*)0);
glEnableVertexAttribArray(0);

/* Loop until the user closes the window */
while (!glfwWindowShouldClose(window)) {

    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);

    ourShader.use();
    glBindVertexArray(VAO);
    glDrawArrays(GL_LINE_STRIP, 0, length);

    /* Swap front and back buffers */
    glfwSwapBuffers(window);

    /* Poll for and process events */
    glfwPollEvents();
}

glDeleteVertexArrays(1, &VAO);
glDeleteBuffers(1, &VBO);
delete[] vertices;
```

```
    glfwTerminate();
    return 0;
}
```

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

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

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

```
#version 330 core
out vec4 FragColor;

void main()
{
    FragColor = vec4(0.0f, 0.0f, 0.0f, 1.0f);
}
```

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` 命令行程序，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` 命令行程序。
`IncrPoly-OpenGL-mac` 图形程序。

A.3 Linux

`IncrPoly-linux` 命令行程序。
`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：

```
mkdir cmake-build
cd cmake-build
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：

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

①https://cmake.org