先要创建一个纹理对象GLuint textureID，glGenTextures(&textureID)

如果有多个纹理，使用glActiveTexture(GL\_Texture+i)，i=1,2,3,…

将创建的纹理对象绑定到一个纹理目标上：glBindTexture(GL\_TEXTURE\_2D,textureID)

设置纹理目标参数：

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR);

将像素信息绑定到纹理上

glTexImage2d(图片信息)

设为glGenerateMipmap(GL\_TEXTURE\_2D)

更新纹理数据

glBindTexture(GL\_TEXTURE\_2D,textureID)

glTexSubImage2D(GL\_TEXTURE\_2D, 0, 0, 0, width, height, GL\_RGBA, GL\_UNSIGNED\_BYTE, pixelBuffer);

显示纹理

编写着色器代码

编译链接做着色器

// 创建和编译着色器程序

GLuint vertexShader = glCreateShader(GL\_VERTEX\_SHADER);

glShaderSource(vertexShader, 1, &vertexShaderSource, nullptr);

glCompileShader(vertexShader);

GLuint fragmentShader = glCreateShader(GL\_FRAGMENT\_SHADER);

glShaderSource(fragmentShader, 1, &fragmentShaderSource, nullptr);

glCompileShader(fragmentShader);

shaderProgram = glCreateProgram();

glAttachShader(shaderProgram, vertexShader);

glAttachShader(shaderProgram, fragmentShader);

glLinkProgram(shaderProgram);

glDeleteShader(vertexShader);

glDeleteShader(fragmentShader);

创建矩形，使用VAO，VBO，EBO

float vertices[16] = {

// 顶点坐标 纹理坐标

1.0f, 1.0f, 1.0f, 0.0f, // 右下角

1.0f, -1.0f, 1.0f, 1.0f, // 右上角

-1.0f, -1.0f, 0.0f, 1.0f, // 左上角

-1.0f, 1.0f, 0.0f, 0.0f // 左下角

};

unsigned int indices[6] = {

0, 1, 3, // 第一个三角形

1, 2, 3 // 第二个三角形

};

glGenVertexArrays(1, &VAO);

glBindVertexArray(VAO);

glGenBuffers(1, &VBO);

glBindBuffer(GL\_ARRAY\_BUFFER, VBO);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(vertices), vertices, GL\_STATIC\_DRAW);

glGenBuffers(1, &EBO);

glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, EBO);

glBufferData(GL\_ELEMENT\_ARRAY\_BUFFER, sizeof(indices), indices, GL\_STATIC\_DRAW);

glVertexAttribPointer(0, 2, GL\_FLOAT, GL\_FALSE, 4 \* sizeof(float), (void\*)0);

glEnableVertexAttribArray(0);

glVertexAttribPointer(1, 2, GL\_FLOAT, GL\_FALSE, 4 \* sizeof(float), (void\*)(2 \* sizeof(float)));

glEnableVertexAttribArray(1);

// 获取着色器中的uniform位置

modelLoc = glGetUniformLocation(shaderProgram, "model");

使用着色器绘制图像，将图像显示在屏幕x,y,大小为w,h

glUseProgram(shaderProgram);

// 渲染矩形

glBindVertexArray(VAO);

texture->Bind();

const float invPixelWidth = 1.0 / pixelWidth;

const float invPixelHeight = 1.0 / pixelHeight;

const float left = 2.0f \* x \* invPixelWidth - 1.0f;

const float right = 2.0f \* (x + w) \* invPixelWidth - 1.0f;

const float bottom = -(2.0f \* y \* invPixelHeight - 1.0f);

const float top = -(2.0f \* (y + h) \* invPixelHeight - 1.0f);

float vertices[] = {

// 顶点坐标 纹理坐标

right, bottom, 1.0f, 0.0f, // 右下角

right, top, 1.0f, 1.0f, // 右上角

left, top, 0.0f, 1.0f, // 左上角

left, bottom, 0.0f, 0.0f // 左下角

};

if (isFlipX)

{

vertices[2] = 0.0;

vertices[6] = 0.0;

vertices[10] = 1.0;

vertices[14] = 1.0;

}

// 设置矩形的索引

unsigned int indices[] = {

0, 1, 3, // 第一个三角形

1, 2, 3 // 第二个三角形

};

// 更新顶点缓冲区数据

glBindBuffer(GL\_ARRAY\_BUFFER, VBO);

glBufferData(GL\_ARRAY\_BUFFER, sizeof(vertices), vertices, GL\_STATIC\_DRAW);

// 更新索引缓冲区数据

glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, EBO);

glBufferData(GL\_ELEMENT\_ARRAY\_BUFFER, sizeof(indices), indices, GL\_STATIC\_DRAW);

// 绘制矩形

glDrawElements(GL\_TRIANGLES, 6, GL\_UNSIGNED\_INT, 0);

glBindVertexArray(0);