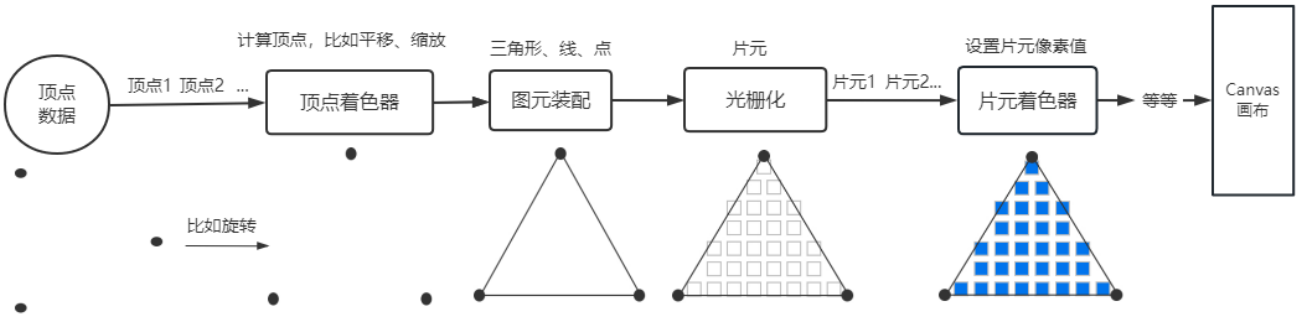
郭隆邦 2023-05-20

9. 片元的屏幕坐标

先回顾下前面关于片元着色器 的讲解。

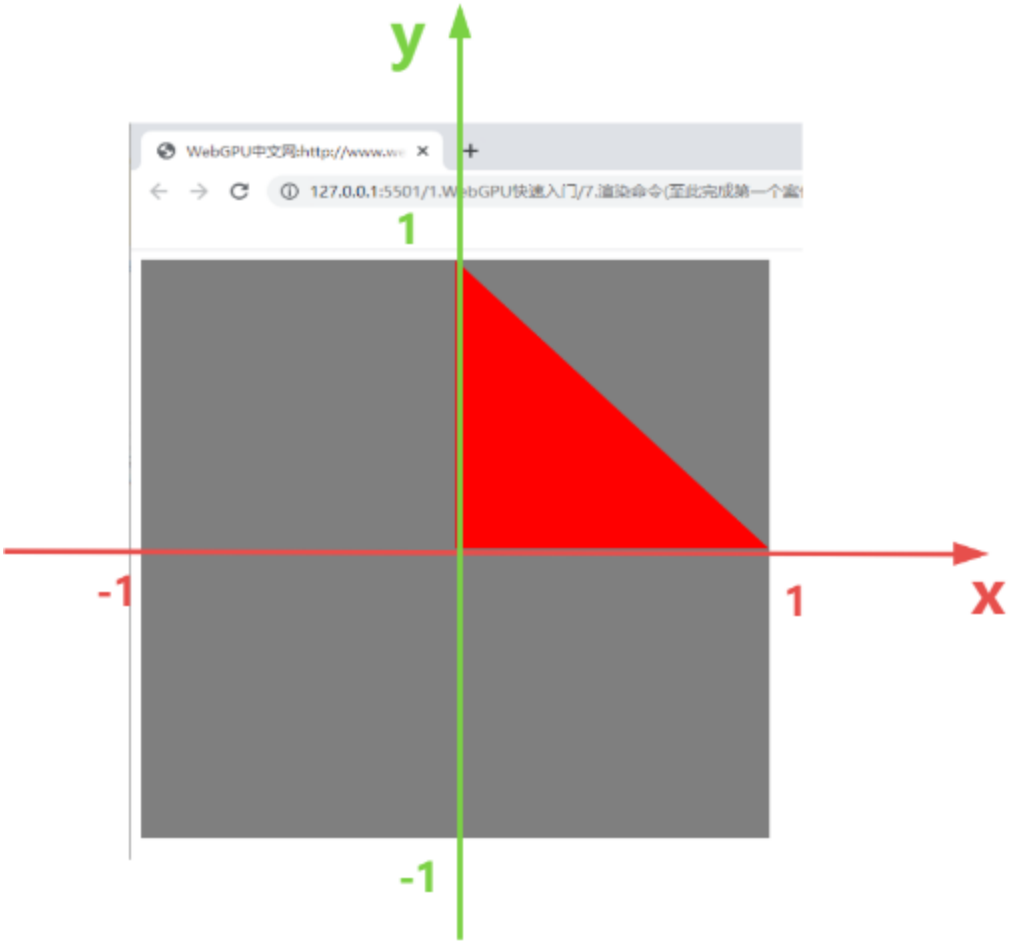


通过WebGPU渲染管线上功能单元光栅化处理，获得几何图形的片元数据。

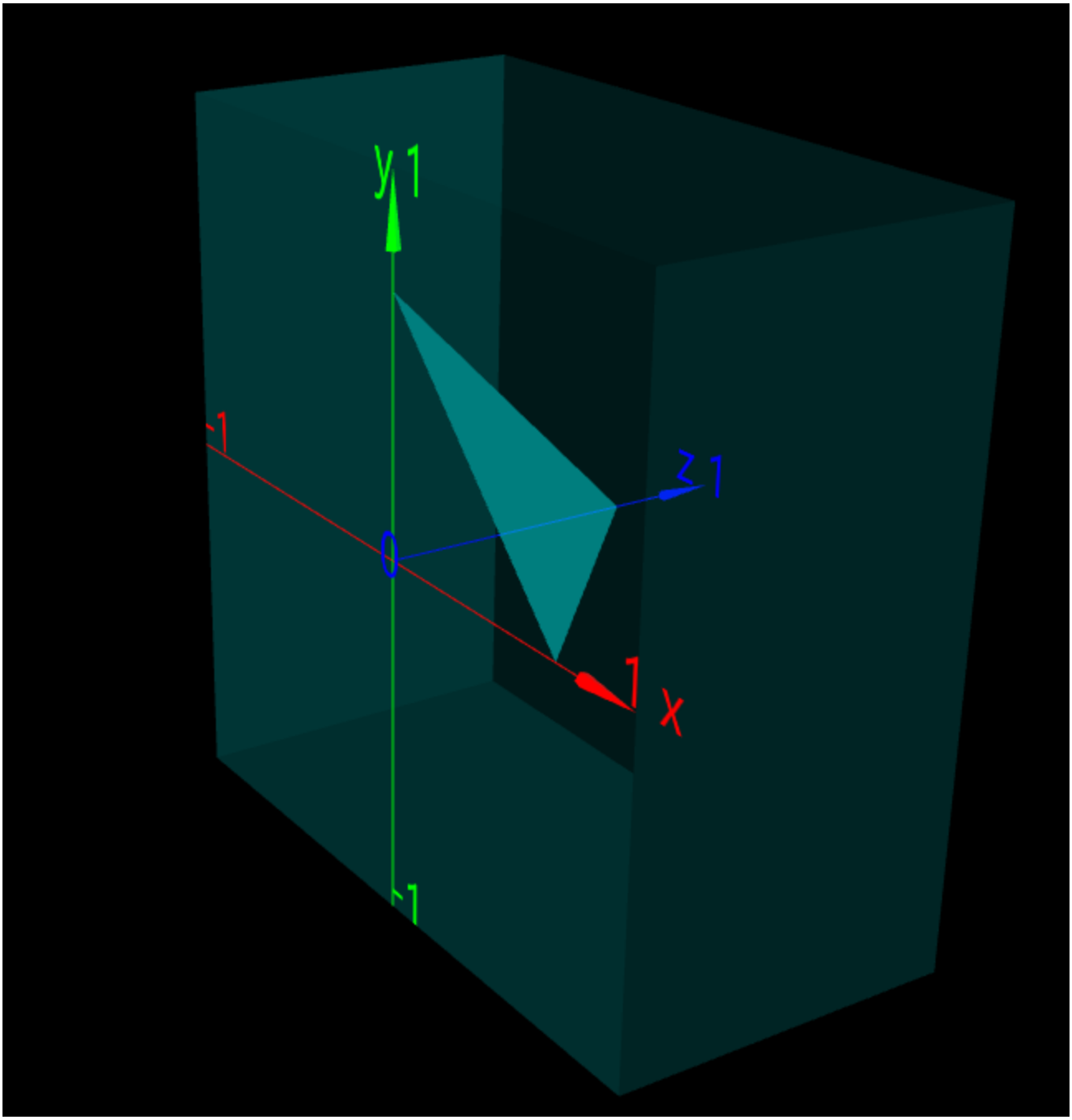
比如一个三角形，经过光栅化，会获取到一个一个片元(像素)填充出来一个三角形的轮廓。

标准设备坐标和屏幕坐标

先回顾下前面讲解的标准设备坐标系

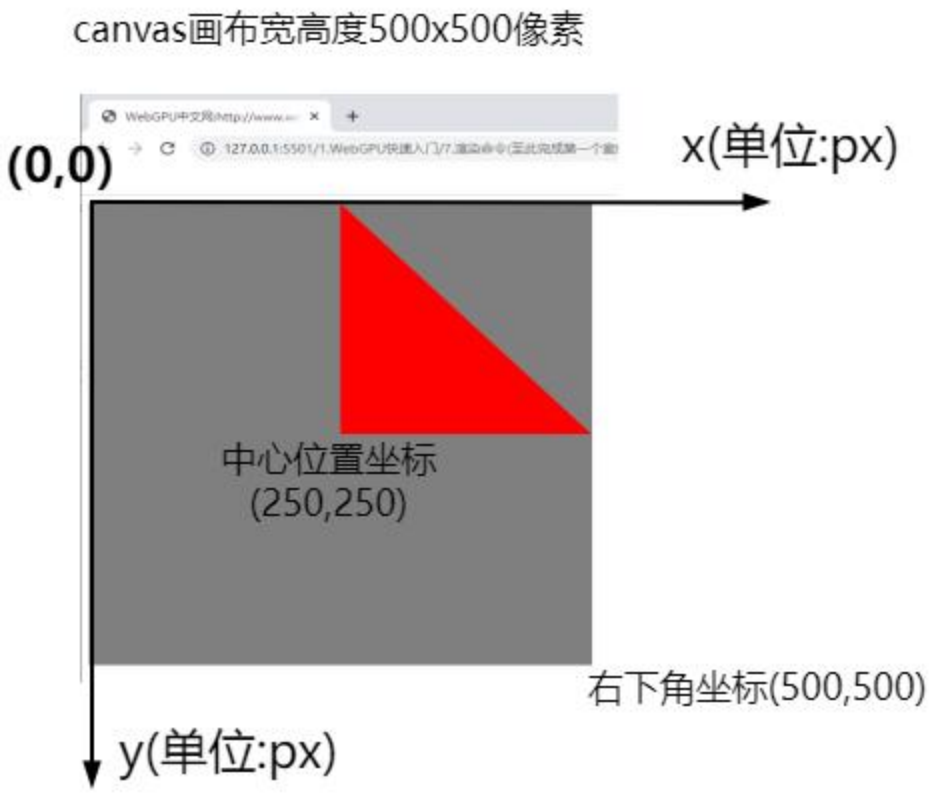


三角形里面的每个片元数据都有一个自己的对应xyz坐标数据，可以使用WebGPU标准设备坐标系去描述片元的3D位置。



在1.8节，讲解过WebGPU投影的知识点，三角形投影到Canvas画布上，片元在canvas画布上的位置，可以使用屏幕坐标系描述。

屏幕坐标系，坐标原点是Canvas画布的左上角，x轴水平向右，y轴竖直向下，x最大值是canvas画布宽度，y最大值是canvas画布的高度，单位是像素值px。



片元着色器获取片元屏幕坐标xy

片元着色器中可以通过片元着色器主函数`main()`的参数获取光栅化后得到的片元坐标数据。

命名一个变量`fragCoord`表示片元坐标，当然你也可以用其它的名字，然后通过内置变量`position`指定`fragCoord`变量表示片元位置数据，具体写法是添加前缀`@builtin(position)`，最后注明数据类型`fragCoord : vec4<f32>`然后。

```js
@fragment
fn main(@builtin(position) fragCoord : vec4<f32>) -> @location(0) vec4<f32> {
    var x = fragCoord.x;//片元屏幕坐标x
    var y = fragCoord.y;//片元屏幕坐标y
}
```

根据屏幕坐标控制片元颜色

canvas画布的宽度是500px，x方向中间屏幕坐标是250，下面代码以250为分界线，设置片元颜色，左边红色，右边绿色。

```js
fn main(@builtin(position) fragCoord : vec4<f32>) -> @location(0) vec4<f32> {
    var x = fragCoord.x;//片元屏幕坐标x
```

```
    if(x < 250.0){
        // 片元x屏幕坐标小于250，片元设置为红色
        return vec4<f32>(1.0, 0.0, 0.0, 1.0);
    }else{
        // 片元x屏幕坐标不小于250，片元设置为绿色
        return vec4<f32>(0.0, 1.0, 0.0, 1.0);
    }
}
```

把中心位置(250,250)左上角设置为红色，其他区域绿色。

```js
@fragment
fn main(@builtin(position) fragCoord : vec4<f32>) -> @location(0) vec4<f32> {
    var x = fragCoord.x;//片元屏幕坐标x
    var y = fragCoord.y;//片元屏幕坐标y
    // 左上角红色，其他区域绿色
    if(x < 250.0 && y < 250.0){
        return vec4<f32>(1.0, 0.0, 0.0, 1.0);
    }else{
        return vec4<f32>(0.0, 1.0, 0.0, 1.0);
    }
}
```

练习题:设置颜色渐变

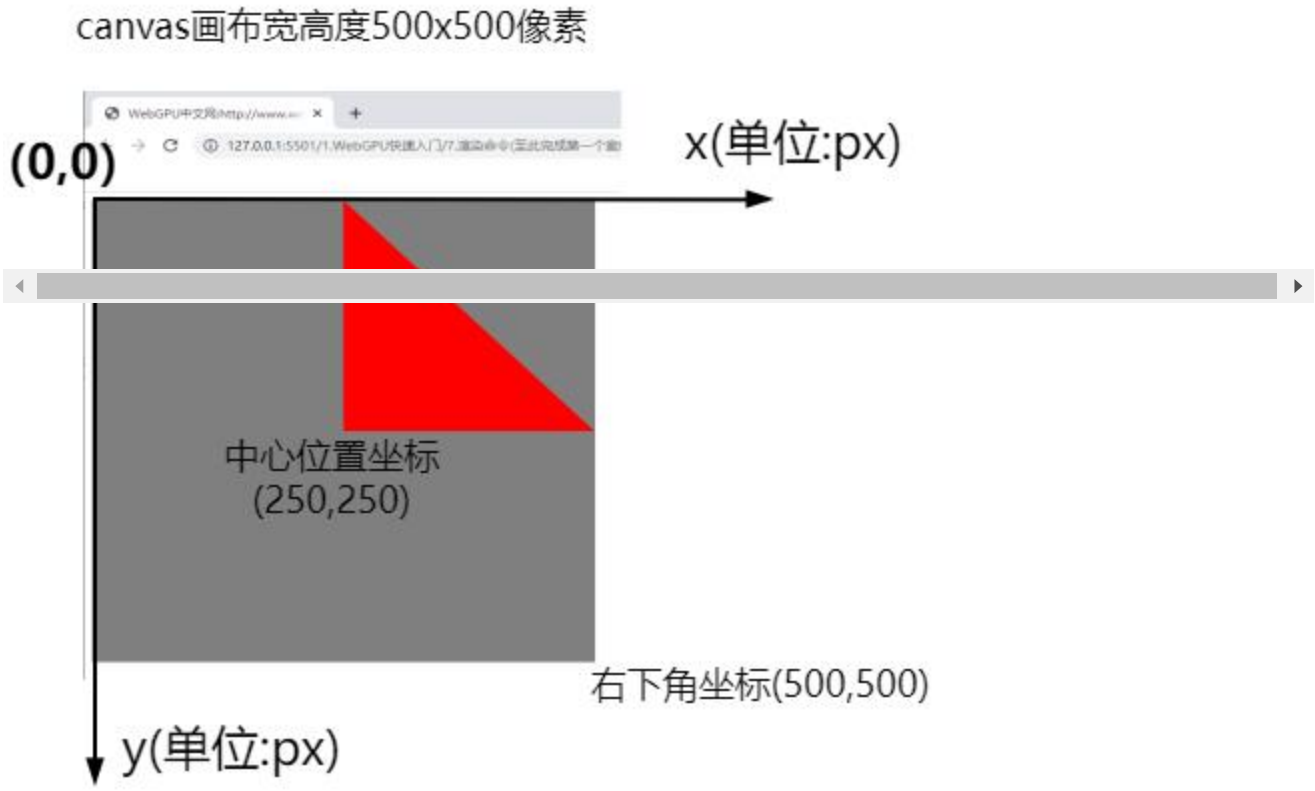
给下面一个三角形设置一个渐变色，左边蓝色，右边红色，中间是两种颜色的过渡色。

```js
// 三角形顶点坐标
1.0, 0.0, 0.0,
0.0, 1.0, 0.0,
0.0, 0.0, 0.0,
```

下面代码是根据片元屏幕坐标x设置渐变色，你可以根据需要自由编写渐变公式。

```js
@fragment
fn main(@builtin(position) fragCoord : vec4<f32>) -> @location(0) vec4<f32> {
    var x = fragCoord.x;//片元屏幕坐标x
    // 渐变色
    var per: f32 = (x-250.0)/250.0;
    return vec4<f32>(per, 0.0, 1.0-per, 1.0);
```

```
}
```



← 8. 绕y轴旋转动画

10. 片元深度值、深度缓冲区→