郭隆邦 2023-05-20

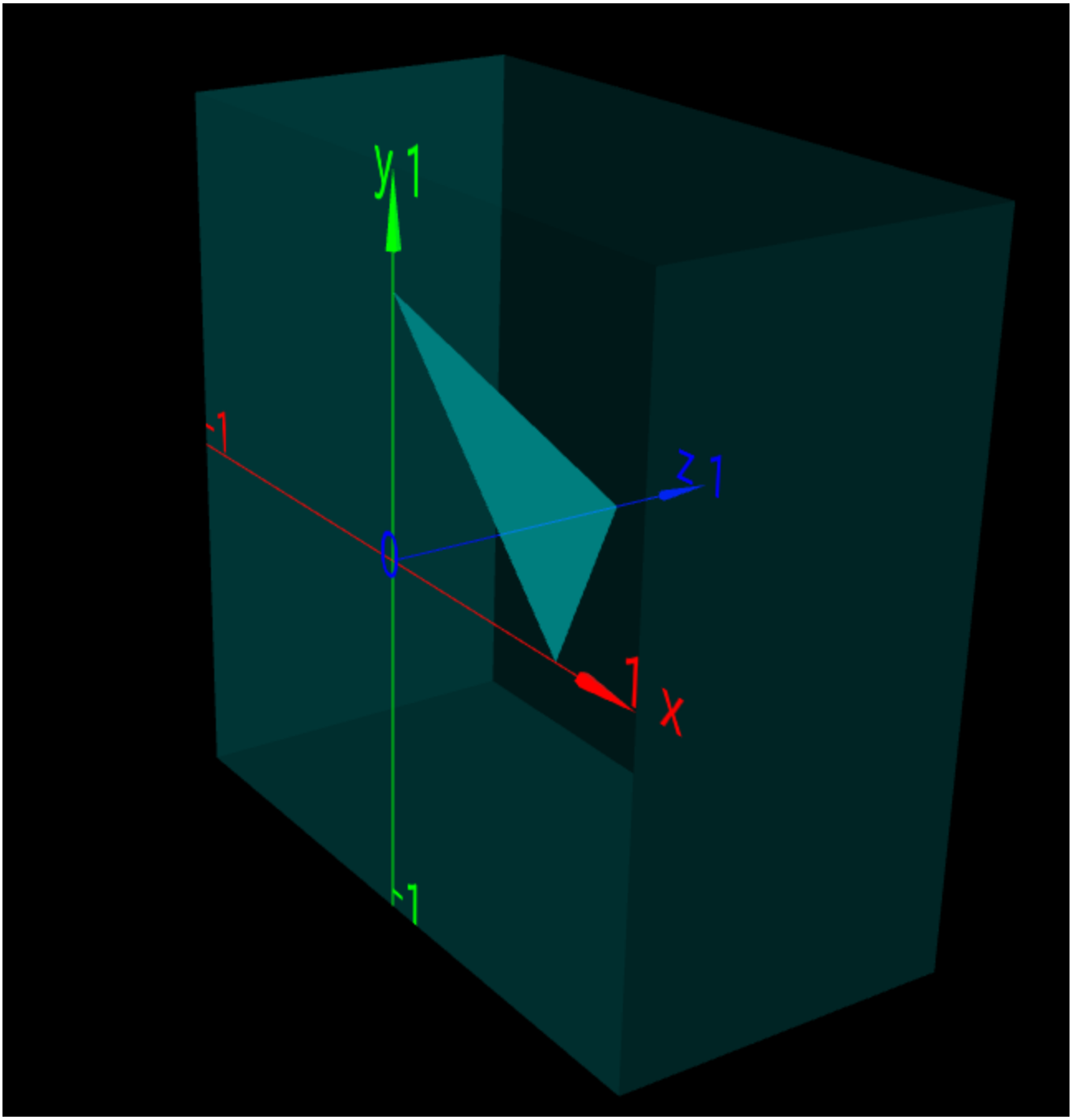
10. 片元深度值、深度缓冲区

接着上节课片元屏幕坐标，给大家介绍下，WebGPU片元深度值(Z坐标)和深度缓冲区的概念。

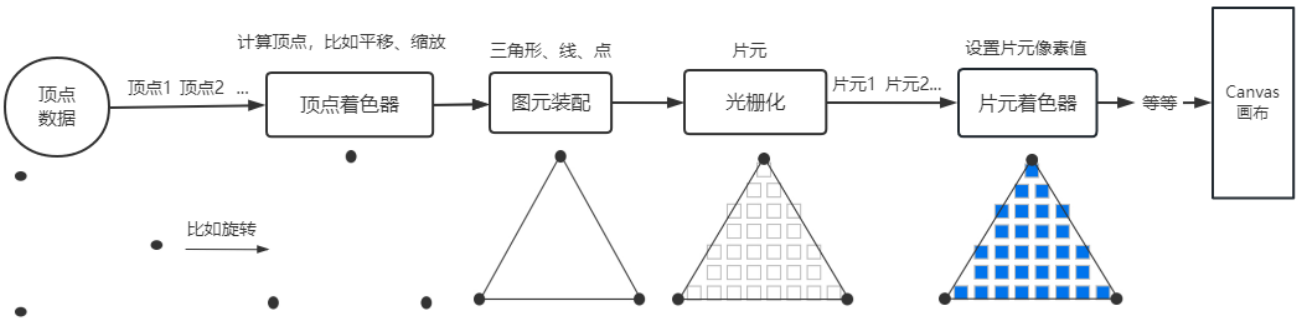
片元深度值(z坐标)

在xyz轴上分别取一个点构成构成一个三角形，然后以该三角形为例给大家讲解片元的深度值。

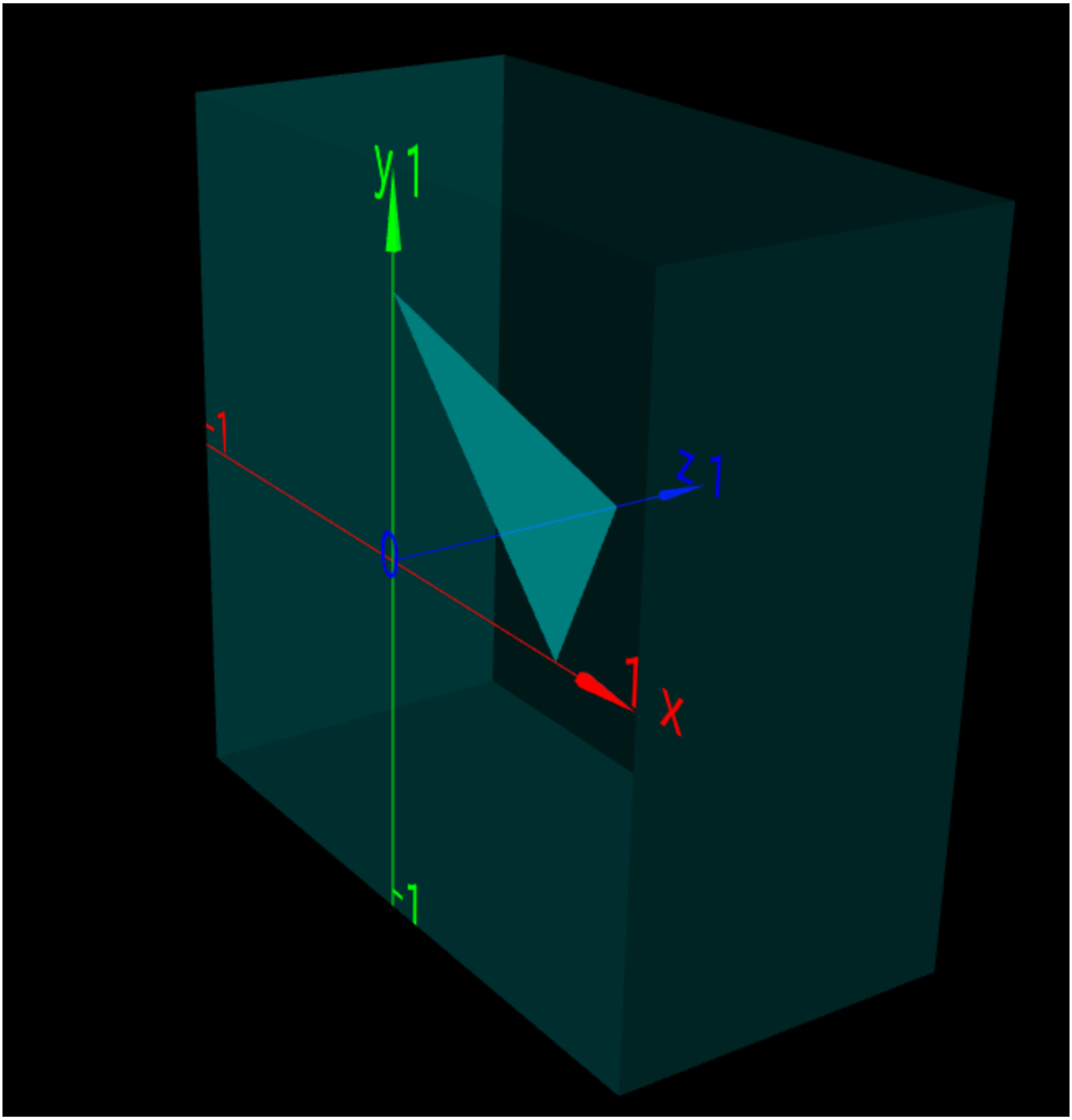
```js
// 三角形顶点坐标
1.0, 0.0, 0.0,
0.0, 1.0, 0.0,
0.0, 0.0, 1.0,
```



经过渲染管线处理后生成片元数据

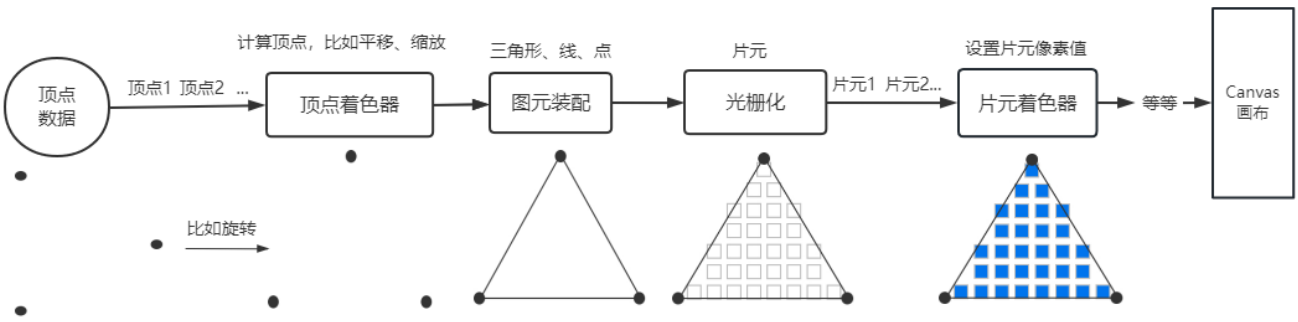


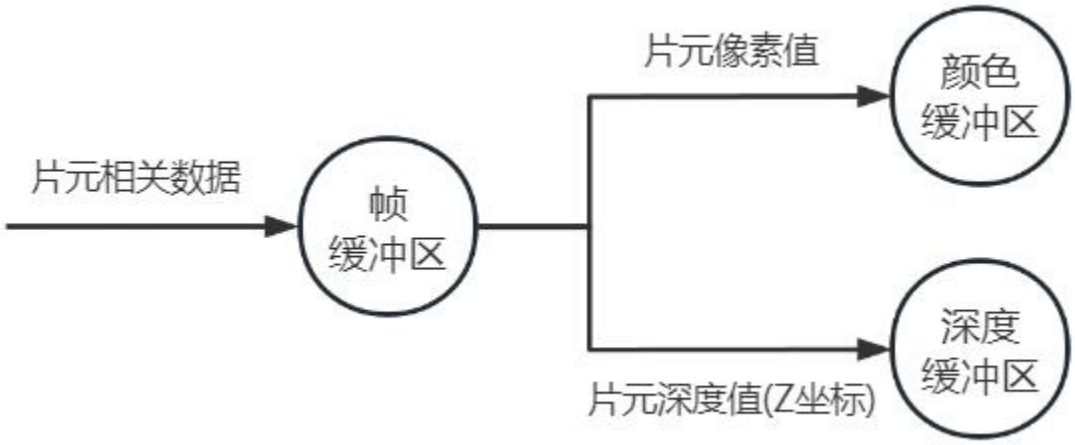
三角形里面的每个片元数据都有一个自己的对应xyz坐标数据,本节课主要给大家讨论下Z坐标，你也可以称为片元深度值。



帧缓冲区简介

经过渲染管线处理，会生成一些片元相关的数据，片元的颜色(像素)数据存储在颜色缓冲区，片元的深度值Z存储在深度缓冲区中，颜色缓冲区、深度缓冲区都是帧缓冲区的一部分。





片元着色器获取片元深度值

获取片元的屏幕坐标xy

```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
}
```

获取片元深度值方式和屏幕坐标一样，也是通过`@builtin(position)`声明的变量获取。

```js
@fragment
fn main(@builtin(position) fragCoord : vec4<f32>) -> @location(0) vec4<f32> {
    //片元深度值，也就是z坐标
    var z = fragCoord.z;
}
```

通过内置变量`@builtin(position)`可以获取片元的坐标，`fragCoord.x`、`fragCoord.y`是屏幕坐标,不过`fragCoord.z`是并不是屏幕坐标，在该案例中，三角形三个顶点位置对应的片元z，还是你在顶点缓冲区中三个顶点的z坐标，其余的片元z值在这三个点的z之间。

根据片元深度值控制片元颜色

```js
@fragment
fn main(@builtin(position) fragCoord : vec4<f32>) -> @location(0) vec4<f32> {
    // 片元深度值
    var z = fragCoord.z;
    if(z < 0.5){
        // 片元深度值小于0.5，片元设置为红色
        return vec4<f32>(1.0, 0.0, 0.0, 1.0);
    }else{
        // 片元深度值不小于250，片元设置为绿色
        return vec4<f32>(0.0, 1.0, 0.0, 1.0);
    }
}
```

片元深度可视化

z的范围0~1，颜色范围绿色(0, 1, 0)到黄色(1, 1, 0)之间变化

```js
@fragment
fn main(@builtin(position) fragCoord : vec4<f32>) -> @location(0) vec4<f32> {
    // 片元深度值
    var z = fragCoord.z;
    // 可视化片元深度
    return vec4<f32>(z, 1.0, 0.0, 1.0);
}
```

z的范围0~1，颜色范围蓝色(0, 0, 1)到红色(1, 0, 0)之间变化

```js
@fragment
fn main(@builtin(position) fragCoord : vec4<f32>) -> @location(0) vec4<f32> {
    // 片元深度值
    var z = fragCoord.z;
    // 可视化片元深度
    return vec4<f32>(z,0.0, 1.0-z, 1.0);
}
```

← 9. 片元的屏幕坐标 11. WebGPU顶点数据插值计算→