郭隆邦 2023-06-02

13. 顶点颜色数据插值计算

前面给大家介绍过**顶点位置**坐标数据，本节课给大家介绍一种新的顶点数据，就是**顶点颜色**数据。

创建顶点颜色数据

顶点位置数据和顶点位置数据的缓冲区。

```js
//创建顶点位置坐标数据
const vertexArray = new Float32Array([
    0.0, 0.0, 0.0,//顶点1 位置坐标
    1.0, 0.0, 0.0,//顶点2 位置坐标
    0.0, 1.0, 0.0,//顶点3 位置坐标
]);
// 创建顶点位置数据的缓冲区
const vertexBuffer = device.createBuffer({
    size: vertexArray.byteLength,
    usage: GPUBufferUsage.VERTEX | GPUBufferUsage.COPY_DST,
});
// 顶点位置数据写入缓冲区
device.queue.writeBuffer(vertexBuffer, 0, vertexArray);
```

类比顶点位置数据，创建顶点颜色数据和顶点颜色数据对应的缓冲区。

```js
//创建顶点颜色数据
const colorArray = new Float32Array([
    1.0, 0.0, 0.0,//顶点1颜色数据 红色
    0.0, 1.0, 0.0,//顶点2颜色数据 绿色
    0.0, 0.0, 1.0,//顶点3颜色数据 蓝色
]);
const colorBuffer = device.createBuffer({// 创建顶点颜色数据的缓冲区
    size: colorArray.byteLength,
    usage: GPUBufferUsage.VERTEX | GPUBufferUsage.COPY_DST,
});
// 顶点颜色数据写入缓冲区
```

```
device.queue.writeBuffer(colorBuffer, 0, colorArray);
```

渲染管线配置顶点颜色数据获取方式

渲染管线关于顶点颜色数据相关的参数设置，可以参考顶点位置数据配置参数。

`buffers`属性的数组元素是对象构成的，一个对象对应一个顶点缓冲区。

```js
// 渲染管线
const pipeline = device.createRenderPipeline({
    vertex: {//顶点相关配置
        ...
        buffers: [//顶点缓冲区相关设置
            {
                arrayStride: 3 * 4,
                attributes: [{
                    //顶点位置缓冲区存储位置标记
                    shaderLocation: 1,
                    format: "float32x3",
                    offset: 0
                }]
            }, {
                //一个顶点的颜色包含rgb三个分量,每个分量4字节
                arrayStride: 3 * 4,
                attributes: [{
                    //顶点颜色缓冲区存储位置标记
                    shaderLocation: 1,
                    format: "float32x3",
                    offset: 0
                }]
            }
        ]
    },
    ...
});
```

渲染通道`.setVertexBuffer()`设置顶点颜色数据

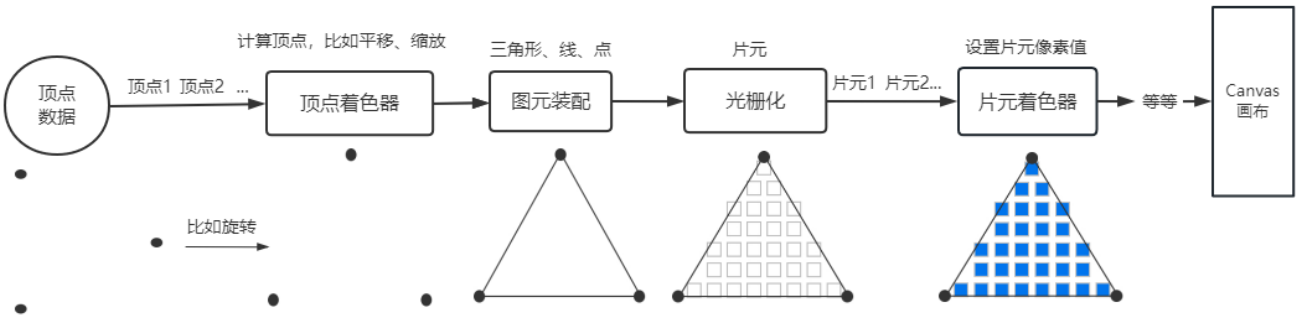
```js
// 渲染通道设置顶点位置数据对应顶点缓冲区    参数一：0表示第1个顶点缓冲区
renderPass.setVertexBuffer(0, vertexBuffer);
// 渲染通道设置顶点颜色数据对应顶点缓冲区    参数一：1表示第2个顶点缓冲区
```

```
renderPass.setVertexBuffer(1, colorBuffer);
```

如果有多个顶点缓冲区，把`device.createBuffer()`的参数一，依次增加即可。

颜色插值介绍

所谓顶点颜色插值，简单说就是WebGPU会自动根据三角形三个顶点的颜色值，插值计算出来所有片元对应的颜色值，如果三个顶点颜色不同，可以看到一种颜色的渐变效果。



WGSL顶点着色器代码

顶点颜色数据需要插值计算，类比前面顶点位置数据的插值计算编写。

```
struct Out {
    @builtin(position) position : vec4<f32>,
    // vColor表示顶点颜色插值后，每个片元对应的颜色数据
    @location(0) vColor: vec3<f32>
}
@vertex
// main函数输入顶点位置数据和顶点颜色数据
fn main(@location(0) pos: vec3<f32>,@location(1) color: vec3<f32>) -> Out {
    var out: Out;
    out.position = vec4<f32>(pos,1.0);
    out.vColor = color;//顶点颜色插值计算
    return out;
}
```

WGSL片元着色器代码

```
@fragment
// 插值后顶点颜色数据，作为函数参数
fn main( @location(0) vColor: vec3<f32>) -> @location(0) vec4<f32> {
    // 插值后顶点颜色数据作为赋值给每个片元
    return vec4<f32>(vColor, 1.0);
}
```

← 12. 练习—顶点位置插值

14. 顶点位置、颜色数据共享缓冲区→