郭隆邦 2023-06-02

14. 顶点位置、颜色数据共享缓冲区

继续上节课顶点颜色数据的讲解，在上节课三角形案例中，顶点位置和顶点颜色数据分别占用一个顶点缓冲区，下面给大家讲解一个新的例子，就是顶点位置和颜色数据共享同一个顶点缓冲区。

知识回顾

一个三角形的顶点位置、顶点颜色数据分别创建一个顶点缓冲区。

```js
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({...});
device.queue.writeBuffer(vertexBuffer, 0, vertexArray);

//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
]);
// 创建顶点颜色数据的缓冲区
const colorBuffer = device.createBuffer({...});
device.queue.writeBuffer(vertexBuffer, 0, vertexArray);
```

顶点位置、顶点颜色数据共享顶点缓冲区

WebGPU中顶点位置、顶点颜色数据共享顶点缓冲区

```js
//创建顶点数据(顶点位置、顶点颜色)
const vertexArray = new Float32Array([
    // 顶点1位置      顶点1颜色(红色)
```

```
    0.0, 0.0, 0.0, 1.0, 0.0, 0.0,
    // 顶点2位置     顶点2颜色(绿色)
    1.0, 0.0, 0.0, 0.0, 1.0, 0.0,
    // 顶点3位置     顶点3颜色(蓝色)
    0.0, 1.0, 0.0, 0.0, 0.0, 1.0,
]);
// 创建顶点数据的缓冲区(顶点位置、顶点颜色)
const vertexBuffer = device.createBuffer({
    size: vertexArray.byteLength,
    usage: GPUBufferUsage.VERTEX | GPUBufferUsage.COPY_DST,
});
// 顶点位置、颜色数据写入缓冲区
device.queue.writeBuffer(vertexBuffer, 0, vertexArray);
```

渲染管线配置

顶点位置和顶点颜色数据分别占用一个顶点缓冲区，对应的渲染管线配置。

```
// 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: 0,
                    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
                }]
            }
        ]
    },
    ...
```

```
});
```

`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，既然顶点位置和顶点颜色数据共享缓冲区，就不用给`buffers`数组新建一个对象子元素了。

```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: 6 * 4,
                attributes: [{
                    //顶点位置缓冲区存储位置标记
                    shaderLocation: 0,
                    format: "float32x3",
                    offset: 0
                },{
                    //顶点颜色缓冲区存储位置标记
                    shaderLocation: 1,
                    format: "float32x3",
                    //一个顶点可能包含多种类型顶点数据，间隔顶点位置的3个数字，才能获取颜
                    offset: 3 * 4
                }]
            }
        ]
    },
    ...
});
```

一个顶点包含位置数据3个分量、颜色数据3个分量，共计6个数字,每个数字4字节

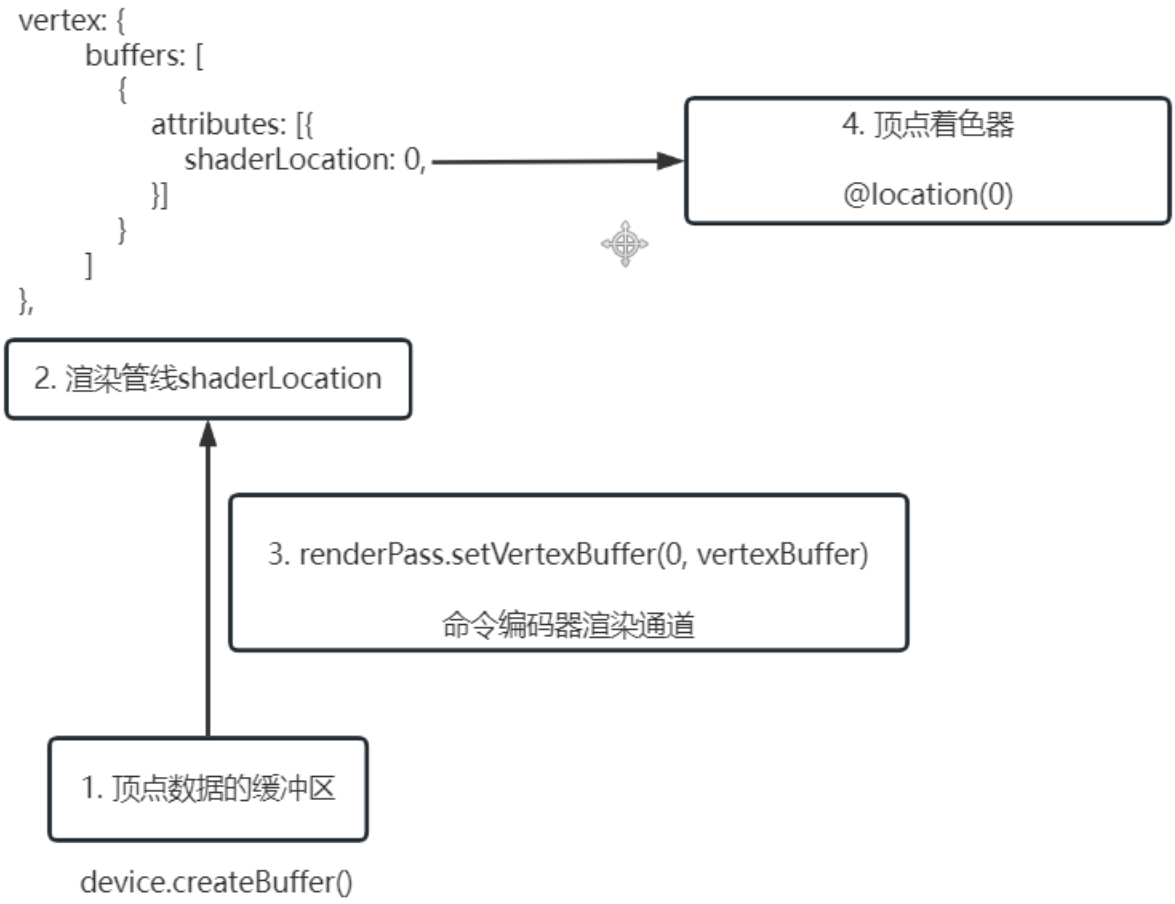
```js
arrayStride: 6 * 4
```

一个顶点可能包含多种类型顶点数据，间隔顶点位置的3个数字，才能获取颜色数据

```js
offset: 3 * 4
```

渲染通道方法`.setVertexBuffer()`详解

前面关于 `.setVertexBuffer()` 的参数1解释比较模糊，下面具体解释下， `.setVertexBuffer()` 的第一个参数准确来说，对应的是渲染管线buffers数组里面元素的索引值。




```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。

前面课程代码会给大家解释， `.setVertexBuffer()` 的参数1和buffers数组中第一个元素对象shaderLocation的值0对应，是因为把buffers数组第一个对象的shaderLocation设置为0,而0刚好是该对象在buffers数组中的索引值，其实你也可以把shaderLocation设置为1、2等其他值，不一定要和 `.setVertexBuffer()` 的参数1一致，shaderLocation的值只要和WGSL顶点着色器代码对应即可。

← 13. 顶点颜色数据插值计算

15. 练习-顶点颜色→