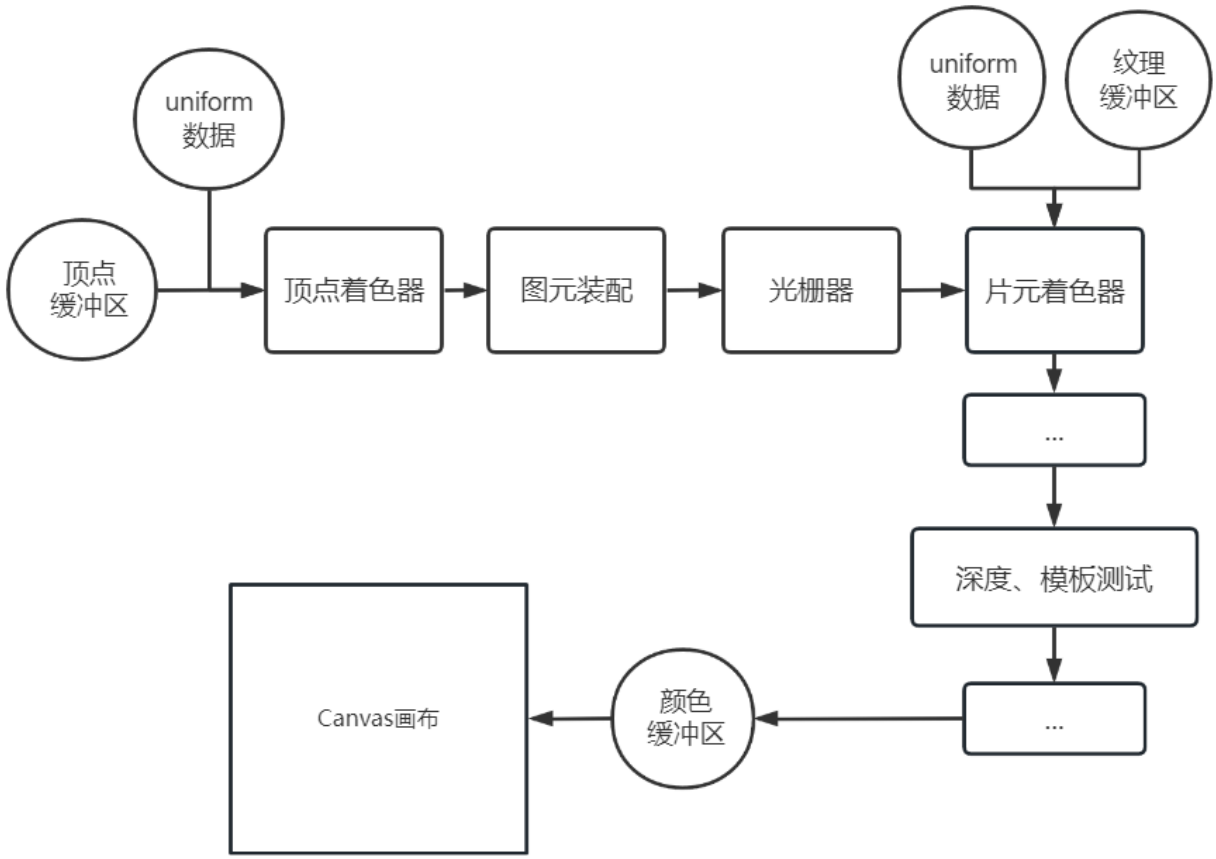
郭隆邦 2023-05-19

5. WebGPU传递uniform数据

前面给大家讲解过，通过JavaScript类型化数组创建顶点数据，然后把顶点数据传递给顶点着色器，本节课给大家讲解如何把uniform数据传递给顶点着色器或者片元着色器。

比如上节课介绍的顶点着色器的缩放矩阵，是写在顶点着色器代码中的，你可以把shader代码中的缩放矩阵数据，放在JavaScript代码中，然后再传入到顶点着色器。



顶点缓冲区知识点回顾

把顶点数据在JavaScript代码中创建传入到顶点着色器中。

```js
//类型化数组表示顶点数据
const vertexArray = new Float32Array([0.0, 0.0, 0.0,1.0, 0.0, 0.0,0.0, 1.0, 0.0,
// 创建存储顶点数据的顶点缓冲区
const vertexBuffer = device.createBuffer({
```

```
    size: vertexArray.byteLength,//顶点数据的字节长度
    //usage设置该缓冲区的用途
    usage: GPUBufferUsage.VERTEX | GPUBufferUsage.COPY_DST,
});
//把vertexArray里面的顶点数据写入到vertexBuffer对应的GPU显存缓冲区中
device.queue.writeBuffer(vertexBuffer, 0, vertexArray);
```

你参考前面入门案例关于顶点数据传递的讲解，来理解本节课uniform传递矩阵数据的传递。

类型化数组表示缩放矩阵数据

上节课讲解的，在顶点着色器中创建了一个缩放矩阵。

```js
@vertex
fn main(@location(0) pos: vec3<f32>) -> @builtin(position) vec4<f32> {
    // 创建一个缩放矩阵(沿着x、y分别缩放0.5倍)
    //0.5  0    0    0
    //0    0.5  0    0
    //0    0    1    0
    //0    0    0    1
    // 矩阵元素一列一列输入mat4x4<f32>()
    var S = mat4x4<f32>(0.5,0.0,0.0,0.0,  0.0,0.5,0.0,0.0,  0.0,0.0,1.0,0.0,  0.
}
```

缩放矩阵数据，从顶点着色器删除，写在JavaScript代码中。

```js
// 创建一个缩放矩阵(沿着x、y分别缩放0.5倍)
//0.5  0    0    0
//0    0.5  0    0
//0    0    1    0
//0    0    0    1
const mat4Array = new Float32Array([
    // 矩阵元素一列一列输入作为类型化数组的参数
    0.5, 0.0, 0.0, 0.0,
    0.0, 0.5, 0.0, 0.0,
    0.0, 0.0, 1.0, 0.0,
    0.0, 0.0, 0.0, 1.0
])
```

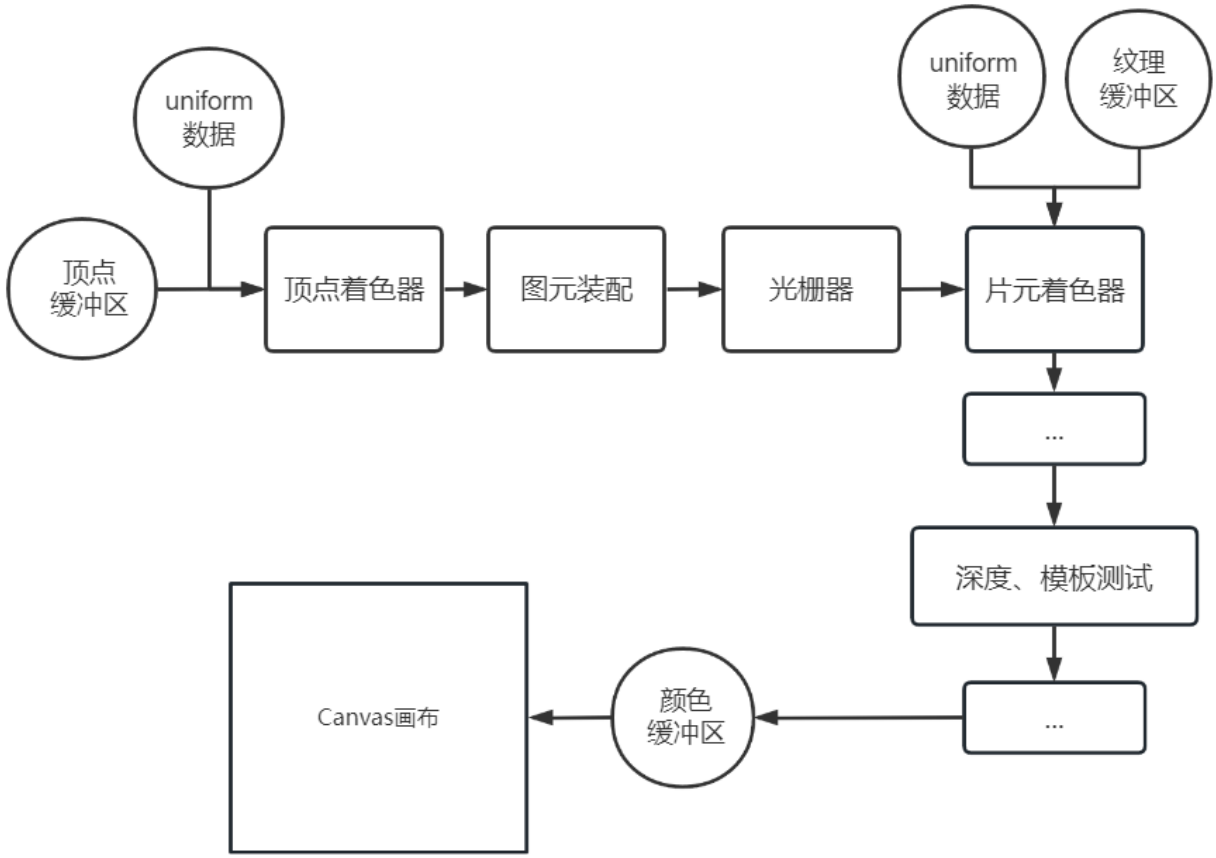
对于这样一个需要传入到着色器中的矩阵数据(非顶点数据)，你可以把他称为uniform数据。

创建uniform数据的缓冲区

创建uniform数据的缓冲区，和创建顶点缓冲区类似，都是使用`.createBuffer()`,其它的代码形式基本一样，注意参数属性usage的值,如果设置为`GPUBufferUsage.VERTEX`表示缓冲区用于存放顶点数据，`GPUBufferUsage.UNIFORM`表示缓冲区用于存放uniform数据(非顶点数据)。

```js
// 创建存储顶点数据的顶点缓冲区
const vertexBuffer = device.createBuffer({
    size: vertexArray.byteLength,//顶点数据的字节长度
    //GPUBufferUsage.VERTEX表示缓冲区用于顶点数据
    usage: GPUBufferUsage.VERTEX | GPUBufferUsage.COPY_DST,
});
```

```js
// 在GPU显存上创建一个uniform数据缓冲区
const mat4Buffer = device.createBuffer({
    size: mat4Array.byteLength,
    usage: GPUBufferUsage.UNIFORM | GPUBufferUsage.COPY_DST
});
```



uniform数据写入到uniform缓冲区中

niform数据写入到uniform缓冲区中，和前面顶点数据传入到顶点缓冲区中，使用的方法一样，都是`device.queue.writeBuffer()`。

```js
// mat4Array里面矩阵数据写入uniform缓冲区mat4Buffer
device.queue.writeBuffer(mat4Buffer, 0, mat4Array)
```

设置uniform数据的绑定组

学习`device.createBindGroup()`的参数，可以类比渲染管线中`.shaderLocation`学习`binding`的属性值，可以设置为0、1、2、3等值进行标记。

```js
// 设置uniform数据的绑定组
const bindGroup = device.createBindGroup({
    layout: pipeline.getBindGroupLayout(0),//绑定组，标记为0
    // 一个组里面可以包含多个uniform数据
    entries: [//每个元素可以用来设置一个uniform数据
        {
```

```
            binding: 0,//标记组里面的uniform数据
            resource: { buffer: mat4Buffer }
        }
    ]
});
```

WGSL着色器代码声明uniform变量

上节课缩放矩阵直接写在顶点着色器中，本节课案例，缩放矩阵写是JavaScript代码中，顶点着色器中只要在顶点着色器代码关键字 `@vertex` 之前，声明一个uniform变量表示缩放矩阵，不用赋值，注意使用 `<uniform>` 标记该变量，系统才会识别为uniform数据，这样才能通过WebGPU API把js代码中顶点数据传入顶点着色器代码中对应的uniform变量。

```js
var<uniform> S:mat4x4<f32>;
@vertex
fn main(@location(0) pos: vec3<f32>) -> @builtin(position) vec4<f32> {
    // 上节课的缩放矩阵S
    // var S = mat4x4<f32>(0.5,0.0,0.0,0.0,  0.0,0.5,0.0,0.0,  0.0,0.0,1.0,0.0,
    return S * vec4<f32>(pos,1.0);//缩放矩阵对顶点缩放变换
}
```

与JavaScript代码中uniform缓冲区中uniform数据关联。

```js
//@group(0)的参数0对应webgpu代码.getBindGroupLayout(0)参数0
//@binding(0)的参数对应webgpu代码.binding的值，保持一致，比如都是0
@group(0) @binding(0) var<uniform> S:mat4<f32>;
@vertex
fn main(@location(0) pos: vec3<f32>) -> @builtin(position) vec4<f32> {
    return S * vec4<f32>(pos,1.0);//缩放矩阵对顶点缩放变换
}
```

```js
const bindGroup = device.createBindGroup({
    layout: pipeline.getBindGroupLayout(0),
    entries: [
        {
            binding: 0,
            resource: { buffer: mat4Buffer }
        }
    ]
```

```
});
```

`.setBindGroup()` 传递uniform数据

```js
// 顶点缓冲区数据和渲染管线shaderLocation: 0表示存储位置关联起来
renderPass.setVertexBuffer(0, vertexBuffer);
```

`.setBindGroup()` 传递uniform数据和 `.setVertexBuffer()` 传递顶点数据的使用方法类似。

```js
// 把绑定组里面的uniform数据传递给着色器中uniform变量
// 参数1的0和.getBindGroupLayout(0)参数一致，都是0
renderPass.setBindGroup( 0, bindGroup );
```

传递一个uniform浮点数数据

声明一个uniform变量，表示三角形顶点坐标x和y方向缩放倍数

```js
@group(0) @binding(0) var<uniform> t:f32;
@vertex
fn main(@location(0) pos: vec3<f32>) -> @builtin(position) vec4<f32> {
    // 创建一个缩放矩阵(沿着x、y分别缩放t倍)
    //t    0    0    0
    //0    t    0    0
    //0    0    1    0
    //0    0    0    1
    // 矩阵元素一列一列输入mat4x4<f32>()
    var S = mat4x4<f32>(t,0.0,0.0,0.0,  0.0,t,0.0,0.0,  0.0,0.0,1.0,0.0,  0.0,0.
    return S * vec4<f32>(pos,1.0);//缩放矩阵对顶点缩放变换
}
```

在JavaScript代码中创建，着色器代码中的uniform变量t对应的数据，就是一个浮点数

```js
// 创建uniform浮点数对应的缓冲区
const t = 0.5;//一个浮点数，表示三角形x和y两个方向的缩放倍数
const mat4Array = new Float32Array([t]);
const mat4Buffer = device.createBuffer({
    size: mat4Array.byteLength,
    usage: GPUBufferUsage.UNIFORM | GPUBufferUsage.COPY_DST
```

```
});
device.queue.writeBuffer(mat4Buffer, 0, mat4Array);
```

练习题：给片元着色器传递颜色数据

```js
// 给片元着色器传递一个颜色数据
const colorArray = new Float32Array([0.0,1.0,0.0]);//绿色
// 在GPU显存上创建一个uniform数据缓冲区
const colorBuffer = device.createBuffer({
    size: colorArray.byteLength,
    usage: GPUBufferUsage.UNIFORM | GPUBufferUsage.COPY_DST
});
// colorArray里面颜色数据写入uniform缓冲区colorBuffer
device.queue.writeBuffer(colorBuffer, 0, colorArray);
```

```js
// 设置uniform数据的绑定组
// 学习createBindGroup的参数，可以类比渲染管线中shaderLocation学习
const bindGroup = device.createBindGroup({
    // .getBindGroupLayout(0)参数0对应shader中@group(0)代码的参数0
    layout: pipeline.getBindGroupLayout(0),//绑定组，标记为0
    // 一个组里面可以包含多个uniform数据
    entries: [//每个元素可以用来设置一个uniform数据
        {
            //binding的值对应@binding(0)的参数，保持一致，比如都是0
            binding: 0,//标记组里面的uniform数据
            resource: { buffer: mat4Buffer }
        },
        {
            //binding的值对应@binding(1)的参数，保持一致，比如都是1
            binding: 1,//标记组里面的uniform数据
            resource: { buffer: colorBuffer }
        }
    ]
});
```

```js
// uniform关键字辅助var声明一个三维向量变量color表示片元颜色
//@binding(1)的参数对应webgpu代码.binding的值，保持一致，比如都是1
@group(0) @binding(1) var<uniform> color:vec3<f32>;
@fragment
fn main() -> @location(0) vec4<f32> {
    // return vec4<f32>(1.0, 0.0, 0.0, 1.0);
    return vec4<f32>(color, 1.0);
```

```
}
```

练习题：传递两个矩阵数据

参考课件案例源码4

← 4. 顶点着色器矩阵变换

6. gl-matrix生成顶点着色器的矩阵→