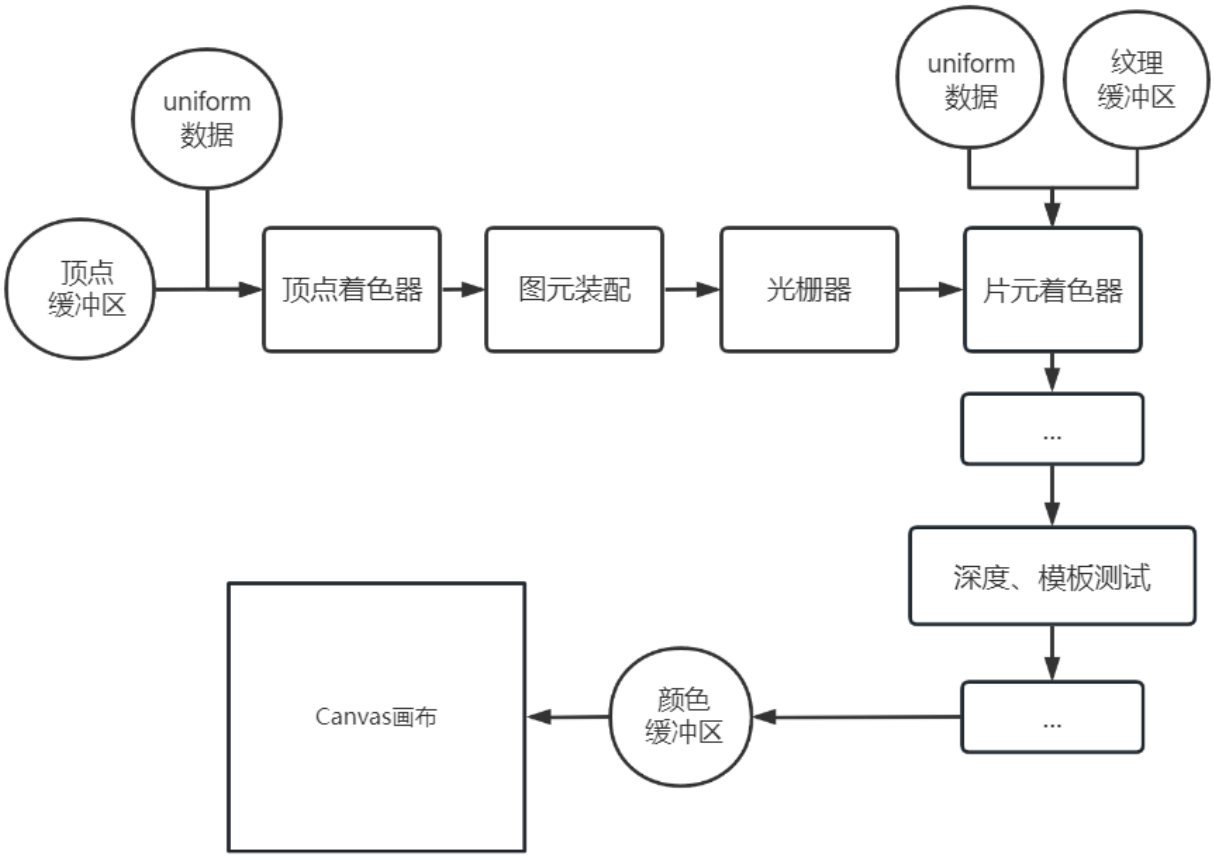
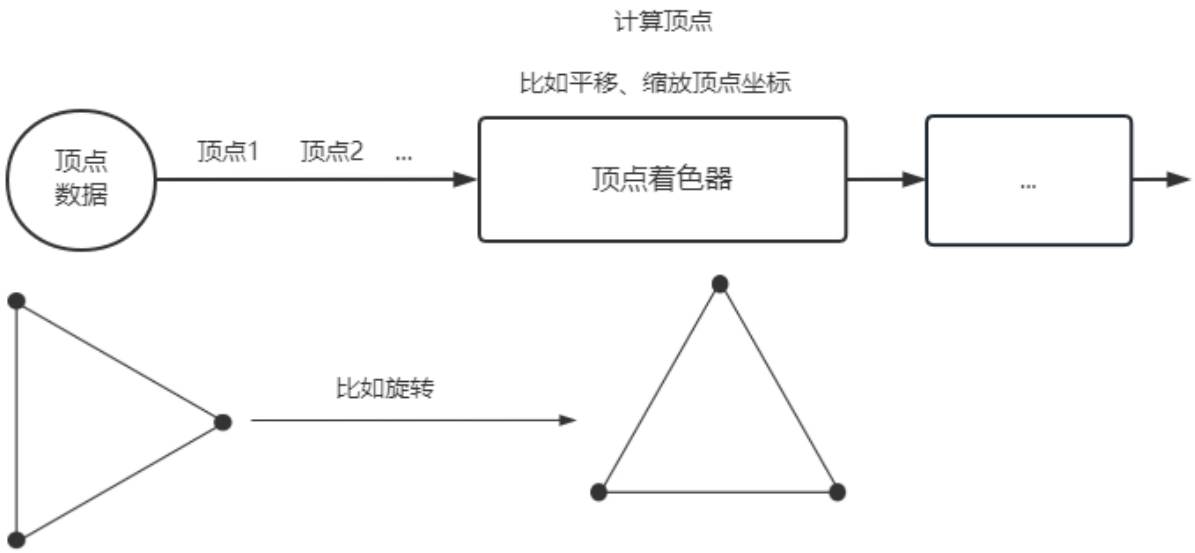
郭隆邦 2023-05-19

4. 顶点着色器矩阵变换

先回顾下1.5节 关于WebGPU渲染管线顶点着色器功能单元的讲解。





下面就给大家演示一个案例，通过顶点着色器对所有的顶点位置坐标，进行缩放变换。

WGSL矩阵语法：矩阵

写WebGPU案例之前，先来熟悉一个WGSL相关的语法，就是特殊数据类型**矩阵**，矩阵的语法可以参考，前面1.4.WGSL语法 关于向量 `vec3` 或 `vec4` 的讲解，基本相似。

2x2矩阵：mat2x2

3x3矩阵：mat3x3

4x4矩阵：mat4x4

```js
// 一个2乘2矩阵
//a  b
//c  d
// 矩阵元素一列一列输入mat2x2<f32>()
mat2x2<f32>(a,c,b,d)
```

WGSL `mat4x4<f32>()` 表示缩放矩阵

`mat4x4<f32>()` 创建一个4x4缩放矩阵(沿着x、y分别缩放0.5倍)。

```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.
}
```

顶点着色器缩放矩阵缩放顶点坐标

WGSL顶点着色器中，4x4缩放矩阵mat4x4对顶点坐标缩放变换。

在WebGPU渲染管线上，所有的顶点坐标，都会被缩放矩阵改变。

```
@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.
    var pos2 = vec4<f32>(pos,1.0);//pos转齐次坐标
    pos2 = S * pos2;//缩放矩阵对顶点缩放变换
    return pos2;
}
```

不声明变量pos2，直接执行`return S * vec4<f32>(pos,1.0);`

```
@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)
    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.
    return S * vec4<f32>(pos,1.0);//缩放矩阵对顶点缩放变换
}
```

平移矩阵平移顶点坐标

```
@vertex
fn main(@location(0) pos: vec3<f32>) -> @builtin(position) vec4<f32> {
    // 创建一个平移矩阵(沿着x、y轴分别平移-1、-1)
    //1   0   0    -1
    //0   1   0    -1
    //0   0   1    0
    //0   0   0    1
    // 矩阵元素一列一列输入mat4x4<f32>()
    var T = mat4x4<f32>(1.0,0.0,0.0,0.0,  0.0,1.0,0.0,0.0,  0.0,0.0,1.0,0.0,  -1
    return T * vec4<f32>(pos,1.0);//平移矩阵对顶点平移变换
}
```

平移和缩放复合变换

写下面代码之前，先回顾下2.2小节 模型矩阵中，关于几何变换顺序对结果的影响。

先平移、后缩放

```
// 先平移、后缩放(矩阵顺序从右往左)
return S * T * vec4<f32>(pos,1.0);
```

先缩放、后平移

```
// 先缩放、后平移(矩阵顺序从右往左)
return T * S * vec4<f32>(pos,1.0);
```

注意矩阵顺序，影响实际变换结果，你可以通过WebGPU的案例代码测试对比。

← 3. gl-matrix数学计算库

5. WebGPU传递uniform数据→