郭隆邦 2023-06-02

12. 练习—顶点位置插值

本节课内容算是一个练习题，并不讲解新的WebGPU知识点，用到的WebGPU知识点是上节课讲解的WebGPU顶点数据插值计算 。

WGSL顶点着色器代码(考虑旋转影响)

参考上节课内容，给2.8小节矩形旋转案例源码增加顶点插值计算shader。

原来顶点着色器代码

```js
@group(0) @binding(0) var<uniform> modelMatrix:mat4x4<f32>;
@vertex
fn main(@location(0) pos: vec3<f32>) -> @builtin(position) vec4<f32> {
    return modelMatrix * vec4<f32>(pos,1.0);
}
```

顶点着色器代码增加顶点插值计算功能

```js
@group(0) @binding(0) var<uniform> modelMatrix:mat4x4<f32>;
struct Out{
    @builtin(position) position:vec4<f32>,
    @location(0) vPosition:vec3<f32>,
}
@vertex
fn main(@location(0) pos: vec3<f32>) -> Out{
    var out:Out;
    out.position = modelMatrix * vec4<f32>(pos,1.0);
    // 顶点插值的时候考虑模型矩阵的影响
    // 注意vPosition数据类型是三维向量，计算结果执行.xyz
    out.vPosition = (modelMatrix * vec4<f32>(pos,1.0)).xyz;
    return out;
}
```

WGSL片元着色器代码

原来的WGSL顶点着色器代码

```js
@fragment
fn main(@location(0) vPosition:vec3<f32>) -> @location(0) vec4<f32> {
    return vec4<f32>(vPosition.x, 0.0, 1.0-vPosition.x, 1.0);
}
```

根据旋转后的顶点插值坐标,设置每个片元的颜色值。

```js
@fragment
fn main(@location(0) vPosition:vec3<f32>) -> @location(0) vec4<f32> {
    // 根据旋转后的顶点插值坐标,设置每个片元的颜色值
    return vec4<f32>(vPosition.z, 0.0, 1.0-vPosition.z, 1.0);
}
```

不考虑模型矩阵对顶点置坐标的影响

不考虑模型矩阵对顶点位置坐标的影响，插值计算的时候，直接设置`out.vPosition = pos;`即可。

```js
fn main(@location(0) pos: vec3<f32>) -> Out{
    var out:Out;
    out.position = modelMatrix * vec4<f32>(pos,1.0);
    // 顶点插值的时候考虑模型矩阵的影响
    // 注意vPosition数据类型是三维向量，计算结果执行.xyz
    // out.vPosition = (modelMatrix * vec4<f32>(pos,1.0)).xyz;
    // 顶点插值的时候不考虑模型矩阵的影响
    out.vPosition = pos;
    return out;
}
```

WGLSL语法练习——vPosition设置为vec4类型

```js
@group(0) @binding(0) var<uniform> modelMatrix:mat4x4<f32>;
struct Out{
    @builtin(position) position:vec4<f32>,
    @location(0) vPosition:vec4<f32>,
}
@vertex
```

```
fn main(@location(0) pos: vec3<f32>) -> Out{
    var out:Out;
    out.position = modelMatrix * vec4<f32>(pos,1.0);
    // 顶点插值的时候考虑模型矩阵的影响
    // 注意vPosition数据类型是三维向量，计算结果执行.xyz
    out.vPosition = modelMatrix * vec4<f32>(pos,1.0);
    return out;
}
```

```
@fragment
fn main(@location(0) vPosition:vec4<f32>) -> @location(0) vec4<f32> {
    return vec4<f32>(vPosition.x, 0.0, 1.0-vPosition.x, 1.0);
}
```

WGLSL语法练习

`out.vPosition` 和 `out.position` 计算方式相同，可以直接把 `out.position` 赋值给 `out.vPosition`

```
@vertex
fn main(@location(0) pos: vec3<f32>) -> Out{
    var out:Out;
    out.position = modelMatrix * vec4<f32>(pos,1.0);
    // out.vPosition = modelMatrix * vec4<f32>(pos,1.0);
    // .vPosition和.position计算方式相同，可以直接把.position赋值给.vPosition
    out.vPosition = out.position;
    return out;
}
```

← 11. WebGPU顶点数据插值计算

13. 顶点颜色数据插值计算→