郭隆邦 2023-04-22

4. 着色器语言WGSL快速了解

WGSL语言是专门给WebGPU定制的着色器语言，就像WebGL OpenGL中使用的GLSL着色器语言。

如果你有GLSL着色器语言基础，那么学习WGSL还是比较容易的。

如果你从未学过其他着色器语言，第一次接触WGSL着色器语言，要想真正理解WGSL，还需要和后面WebGPU具体案例结合讲解，如果只是把语法给你念一遍，过于抽象，很难理解。

文档资料

WGSL英文文档：https://www.w3.org/TR/WGSL/

WebGPU引擎Orillusion团队翻译：https://www.orillusion.com/zh/wgsl.html

学习基础与难度

如果你有其他着色器语言基础，比如GLSL，那么你学习WGSL将会非常简单。

如果没有学过着色器代码，但是有其它静态类型语言的基础，比如Typescript、C等，理解WGSL基础语言也会相对容易。

如果上面两个基础都没有，只是熟悉JavaScript，相对难度高些。

WGSL学习方法

WGSL虽然类似Typescript、C等语言，但是WGSL主要在GPU上执行，有自身的特殊性，结合WebGPU案例学习WGSL语法，才能更好的理解。

所以本节课不做过多的WGSL语法讲解，随意举几个案例，初步了解一些WGSL最基础的语法，你也不用写代码，跟着视频过一遍即可。

WGSL基础类型

下面简单列举了部分WGSL的数据类型

符号	数据类型
bool	布尔
u32	无符号整数
i32	有符号整数
f32	32位浮点数
f16	16位浮点数

`var` 关键字声明变量

WGSL中可以用 `var` 关键字声明变量。

```js
// var关键字声明一个变量a，数据类型是无符号整数
var a:u32;
a = 2;
```

```js
// var关键字声明一个变量a，数据类型是32位浮点数
var a:f32;
a = 2.0;
```

声明的时候直接赋值

```js
// var关键字声明一个32位浮点数
var a:f32 = 2.0;
```

有时候你看别人的WGSL代码，声明变量如果赋值了，可能会省略数据类型标注，这时候WGSL会根据变量的值自动推荐变量的数据类型

```js
var a = 2.0;//推断为f32
```

```js
var a = 2;//推断为i32
```

```js
var a = false;//推断为布尔值
```

变量简单运算

两个变量进行运算，需要保持一样数据类型，否则报错。

```js
// 32位浮点数相加
var a:f32 = 2.0;
var b:f32 = 4.0;
var c:f32 = a+b;
```

```js
// 无符号整数相加
var a:u32 = 2;
var b:u32 = 4;
var c:u32 = a+b;
```

声明函数的关键字 `fn`

```js
// 这还能混合写，牛逼了
fn 函数名( 参数1:数据类型, 参数2:数据类型...){
    // 代码
}
```

```js
fn add( x: f32, y:f32){
    var z: f32 =  x + y;
}
```

如果函数有返回值设置符号 `->` ,后面注明返回值的数据类型

```js
// 这还能混合写，牛逼了
fn 函数名( 参数1, 参数2...) -> 返回值数据类型 {
    return 返回值;
}
```

```js
fn add( x: f32, y:f32) -> f32 {
    return x + y;
```

```
}
// 类比JavaScript语言函数
function add(x , y){
    return x + y;
}
// 类比TypeScript语言函数
function add(x: number, y: number): number {
  return x+y
}
```

if、for等语句

在WGSL中，if、for等语句，和JavaScript逻辑上基本差不多，区别就是注意数据类型即可。

WGSLfor循环语句，基本逻辑

```js
var n:u32 = 10;
var s:f32 = 0.0;
for (var i:u32= 0; i < n; i++) {
    s += 0.05;
}
```

```js
var s:bool;
var a:f32 = 2.0;
if(a>1.0){
    s = true;
}else{
    s = false;
}
```

向量表示颜色

在WGSL中，向量可以表示多种数据，也能进行多种数学运算，咱们这里先不讲解那么多，说些简单的。

```js
// 四维向量有四个分量，可以用来表示颜色的R、G、B、A
var color:vec4<f32> = vec4<f32>(1.0, 0.0, 0.0, 1.0);//红色不透明
```

```js
// 省略:vec4<f32>数据类型
var color = vec4<f32>(1.0, 0.0, 0.0, 1.0);
```

```js
// 先声明一个四维向量变量，再赋值
var color:vec4<f32>;
color = vec4<f32>(1.0, 0.0, 0.0, 1.0);
```

向量表示位置

三维向量 `vec3<f32>` 表示具有三个分量，可以用来表示顶点的xyz坐标。

```js
var pos:vec3<f32>;
pos= vec3<f32>(1.0, 2.0, 3.0);
```

四维向量表示齐次坐标，在WGSL中，表示一个坐标的的时候，如果用四维向量表示，最后一个分量是1.0。改坐标表示xyz的齐次坐标。

```js
var pos:vec4<f32>;
pos= vec4<f32>(1.0, 2.0, 3.0,1.0);
```

一个三维向量转化为四维向量

```js
var pos:vec3<f32>;
pos = vec3<f32>(1.0, 2.0, 3.0);
//等价于vec4<f32>(1.0, 2.0, 3.0,1.0)
var pos2 = vec4<f32>(pos,1.0);
```

一个二维向量转化为四维向量

```js
var pos:vec2<f32>;
pos = vec2<f32>(1.0, 2.0);
//等价于vec4<f32>(1.0, 2.0, 3.0,1.0)
var pos2 = vec4<f32>(pos, 3.0,1.0);
```

结构体

WGSL结构体有点类似JavaScript里面的类

```js
// 定义一个结构体表示点光源
struct pointLight {
    color: vec3<f32>,//光源颜色
    intensity: f32//光源强度
};
```

通过结构体生成一个光源，类似JavaScript中类执行new实例化一个对象。

```js
var light1:pointLight;
light1.color = vec3<f32>(1.0, 0.0, 0.0);
light1.intensity = 0.6;
```

WGSL代码注释

WGSL代码注释和JavaScript语言的习惯一样。

- 单行注释符号 `//`
- 快级注释符号 `/* */`

WGSL语句结尾分号

在JavaScript中，代码语句结尾的分号可以省略，但是WGSL中分号不能省略。

```js
var a:f32 = 2.0;
var a:f32 = 4.0//分号省略，会报错
```

← 3. 创建顶点缓冲区、渲染管线

5. 顶点着色器→