郭隆邦 2023-04-22

2. WebGPU API和Canvas画布

WebGPU提供很多相关的API，通过这些WebGPU API可以控制你的显卡GPU渲染3D场景或计算数据。

WebGPU API文档：https://www.w3.org/TR/webgpu/

GPU概念解释

所谓GPU就是图形处理器，再具体点说，就是你电脑上的显卡，如果为了追求更好的性能，一般会在电脑上安装独立显卡。

GPU设备对象

创建GPU设备对象`device`非常简单，执行`navigator.gpu.requestAdapter()`和`adapter.requestDevice()`两步操作即可完成。

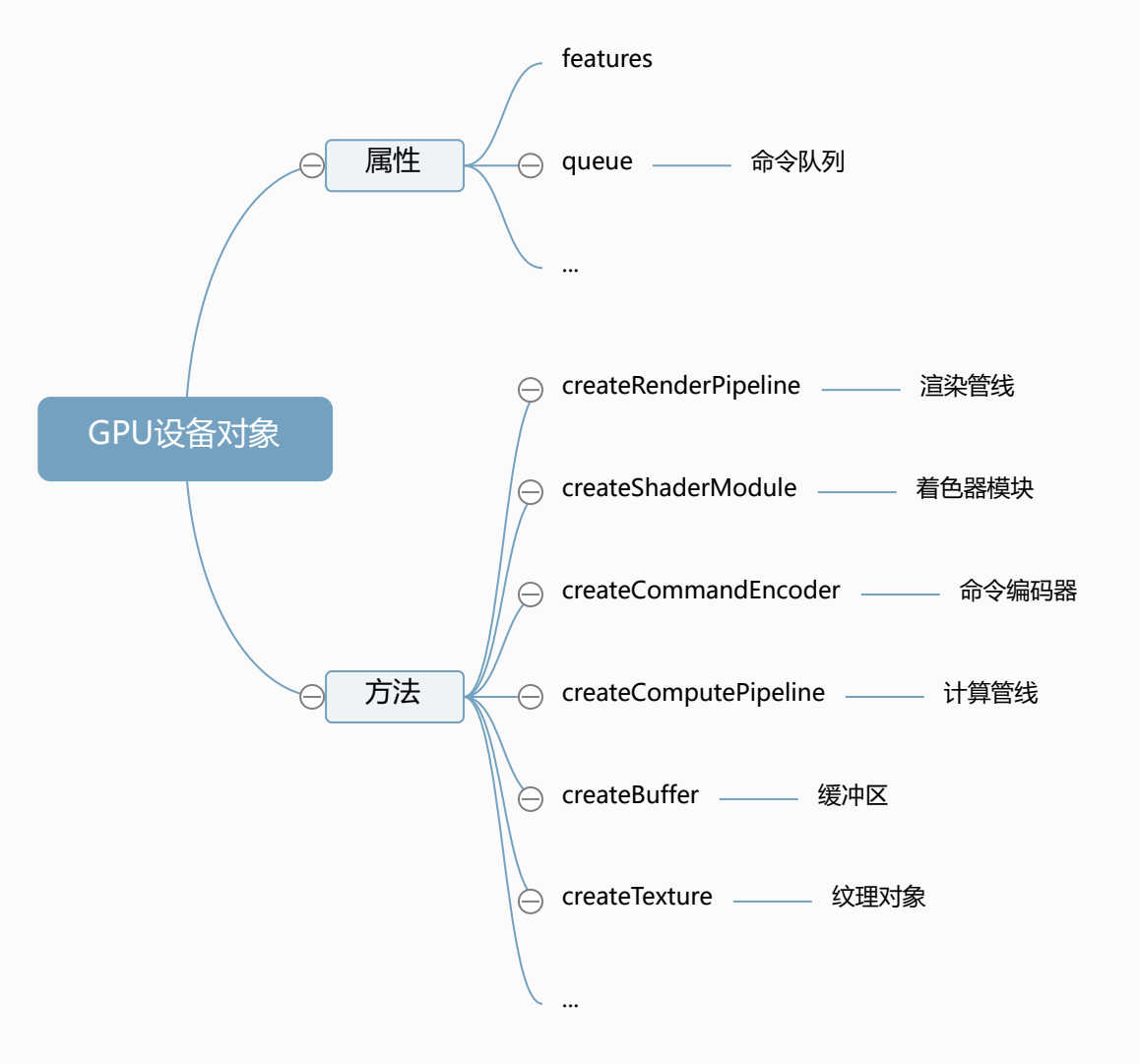
`.requestAdapter()`和`.requestDevice()`都是异步函数，函数前需要加上es6语法的关键字`await`。

```js
// 浏览器请求GPU适配器
const adapter = await navigator.gpu.requestAdapter();
// 获取GPU设备对象，通过GPU设备对象device的WebGPU API可以控制GPU渲染过程
const device = await adapter.requestDevice();
```

浏览器控制台测试查看，适配器对象`adapter`和GPU设备对象`device`对象的一些属性和方法

```js
console.log('适配器adapter',adapter);
console.log('GPU设备对象device',device);
```

GPU设备对象`device`的属性和方法



借助GPU设备对象`device`提供的很多属性和方法，这些属性和方法都是WebGPU API的一部分。后面课程会给大家逐步讲解，如何通过GPU设备对象`device`提供的这些WebGPU API渲染3D场景。

```js
device.createRenderPipeline()//创建渲染管线
device.createComputePipeline()//创建计算管线
device.createShaderModule()//创建着色器模块
device.createCommandEncoder()//创建命令对象(绘制和计算命令)
device.createBuffer()//创建缓冲区对象
...
```

初次接触，可能你还不理解这些WebGPU API，那也没关系，后面会详细讲解，现在你可以随意调用两个API写代码，提前熟悉下。

```js
// 创建渲染管线
const pipeline = device.createRenderPipeline();
// 创建GPU命令对象
```

```
const commandEncoder = device.createCommandEncoder();
```

这也是为什么我们要执行`device = await adapter.requestDevice()`创建GPU设备对象`device`，只有通过创建GPU设备对象，我们才可以获得这些API。

Canvas画布

Canvas画布是一个比较特殊的HTML元素，主要用来实现图形绘制的功能，可以进行2D绘图，可以用来实现WebGL，也可以把WebGPU渲染的图像输出到Canvas画布。

```html
<!-- canvas：用来展示WebGPU渲染的结果 -->
<canvas id="webgpu" width="500" height="500"></canvas>
```

配置WebGPU上下文(Canvas元素作为WebGPU的画布)

获取id名为`webgpu`的Canvas画布元素。

```js
const canvas = document.getElementById('webgpu');
```

Canvas画布对象有一个获取上下文的方法`.getContext()`,参数可以是2d、webgl、webgpu，不同参数用于不同的功能，咱们这里是用于WebGPU渲染，所以参数设置为webgpu。

```js
const context = canvas.getContext('webgpu');
```

通过方法`context.configure()`配置从Canvas画布获取的WebGPU上下文对象`context`。

用人话说就是关联Canvas画布和GPU设备对象`device`,这样就能把Canvas元素作为WebGPU的画布,用来呈现3D渲染效果。

```js
context.configure({
    device: device,//WebGPU渲染器使用的GPU设备对象
});
```

`format`属性和颜色格式有关，如果没有特别需要，可以设置为`navigator.gpu.getPreferredCanvasFormat()`即可，初学可以不用深究。

```js
const format = navigator.gpu.getPreferredCanvasFormat();//获取浏览器默认的颜色格式
context.configure({
    device: device,
    format: format,//颜色格式
});
```

配置WebGPU上下文代码。

```html
<body>
    <!-- canvas：用来展示WebGPU渲染的结果 -->
    <canvas id="webgpu" width="500" height="500"></canvas>
    <script type="module">
        // 1. 初始化WebGPU
        const adapter = await navigator.gpu.requestAdapter();
        // 获取GPU设备对象，通过GPU设备对象device的WebGPU API可以控制GPU渲染过程
        const device = await adapter.requestDevice();

        //配置WebGPU上下文，把id名为webgpu的Canvas元素作为WebGPU的画布
        const canvas = document.getElementById('webgpu');
        const context = canvas.getContext('webgpu');
        const format = navigator.gpu.getPreferredCanvasFormat();//获取浏览器默认的
        context.configure({
            device: device,//WebGPU渲染器使用的GPU设备对象
            format: format,//WebGPU渲染器使用的颜色格式
        });
    </script>
</body>
```

← 1. WebGPU学习开发环境配置

3. 创建顶点缓冲区、渲染管线→