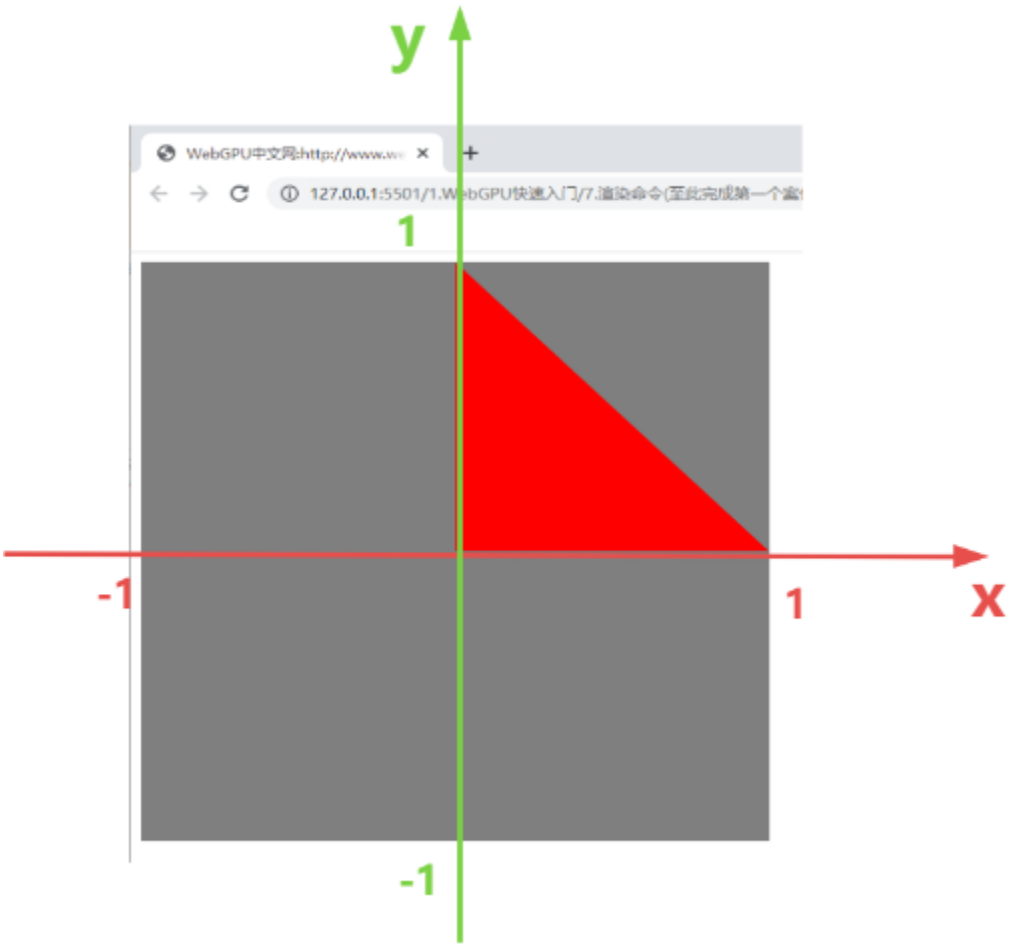
郭隆邦 2023-04-22

3. 创建顶点缓冲区、渲染管线

如果你想渲染一个物体，需要先通过顶点坐标来定义该物体的几何形状，本节课就给大家讲解，怎么通过WebGPU的顶点缓冲区来创建顶点数据。

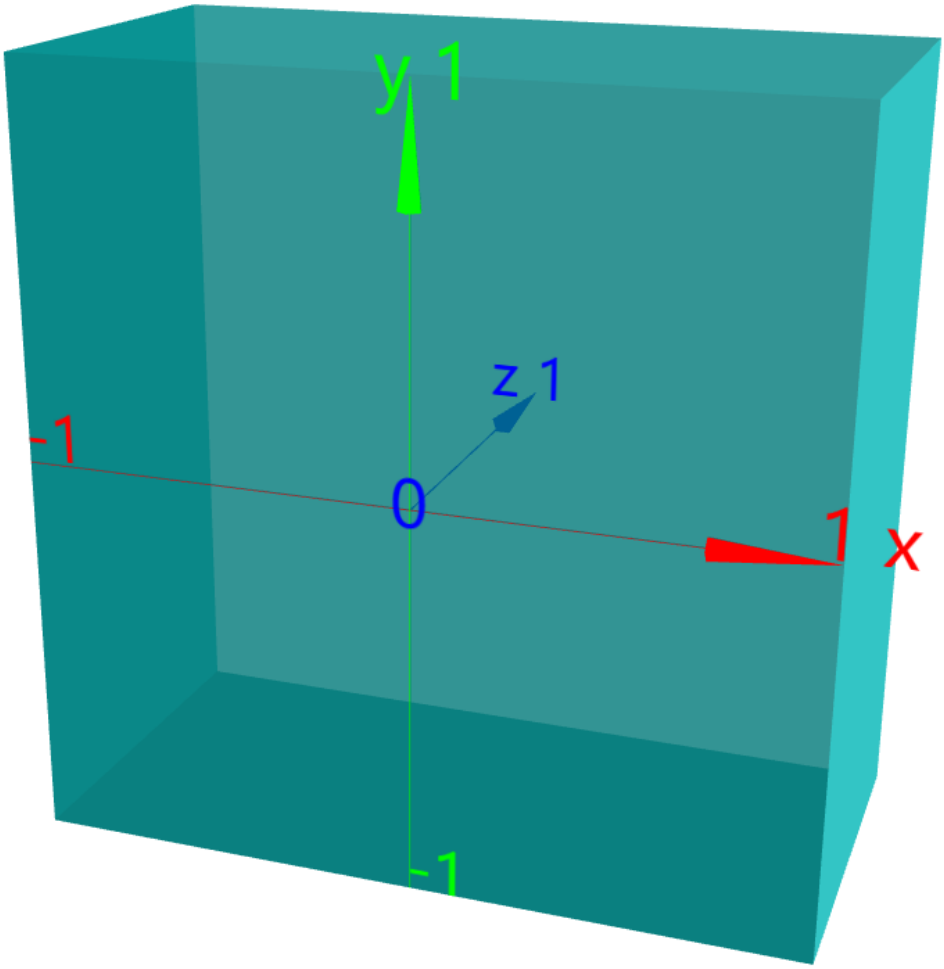
WebGPU坐标系

WebGPU坐标系在Canvas画布上的**坐标原点**是Canvas画布的中间位置，**x轴**水平向**右**，**y轴**竖直向**上**，**z轴**垂直与Canvas画布，朝向屏幕内。



前端开发时候，HTML元素的宽高很多时候是选择以像素为基准定义，比如宽度500px。WebGPU中顶点坐标的表示值采用的是相对值,比如x和y的坐标范围都是[-1,1]，z坐标的范围是[0,1]。

在咱们入门的第一个案例中，先不深入谈WebGPU坐标系，你能先用x、y两个分量绘制一个2D平面图就行，后面涉及到3D效果的时候，再详细展开讲解z坐标、投影矩阵、视图矩阵、模型矩阵等深入概念。



JavaScript类型化数组

类型化数组 文档：https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Typed_arrays

JavaScript类型化数组不同于普通的数组，类型化数组，就是数组的元素可以设置数字的类型，比如浮点数、无符号整数....

实际开发顶点数据往往都比多，在WebGL、WebGPU、threejs等代码中，会用类型化数组类型化数组 表示定义顶点数据。

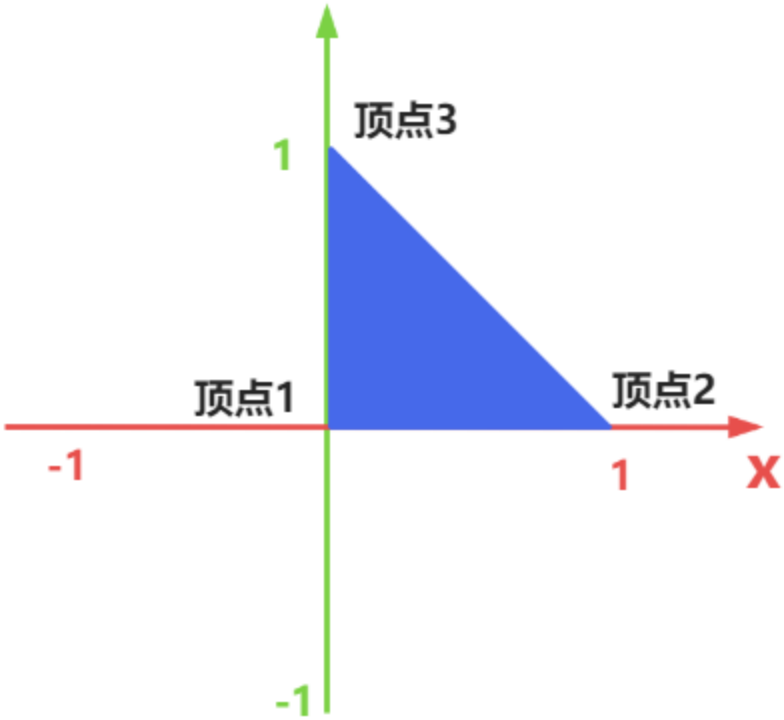
类型化数组 `Float32Array` 表示顶点坐标

一般来说通过WebGPU绘制一个几何图形的时候，比如一个三角形、一个矩形、一个立方体...需要使用顶点先表示几何体的形状。

刚入门，先定义一个简单的几何图形，比如我使用三个顶点的xyz坐标表示一个三角形。实际开发的时候，你可以根据需要，创建任意个顶点坐标数据，来表达一个复杂的几何图案。

类型化数组 `Float32Array` 参数数组里面的元素三个为一组表示顶点的xyz坐标。

```js
const vertexArray = new Float32Array([
    // 三角形三个顶点坐标的x、y、z值
    0.0, 0.0, 0.0,//顶点1坐标
    1.0, 0.0, 0.0,//顶点2坐标
    0.0, 1.0, 0.0,//顶点3坐标
]);
```



创建顶点缓冲区 `.createBuffer()`

通过GPU设备对象的.createBuffer() 方法可以创建一个顶点缓冲区。

关于顶点缓冲区，给大家简单解释下。大家都知道数据，会占用电脑的内存，对于顶点数据而言，同样需要占用电脑内存空间，你可以这么理解，当 `device.createBuffer()` 执行的时候，会在你的电脑显卡GPU的内存(显存)中开辟一片存储空间，用来存存储顶点数据，你可以把这个开辟的存储空间，称为**顶点缓冲区**。

```js
const vertexBuffer = device.createBuffer();
```

缓冲区存储字节长度设置 `size`

设置存储空间的size属性，表示存储空间的大小size。

```js
const vertexBuffer = device.createBuffer({
    size: vertexArray.byteLength,//数据字节长度
});
```

```js
//类型化数组Float32Array一个数字元素，占用存储空间4字节，9个浮点数，数据字节长度9*4
console.log('类型化数组数据字节长度',vertexArray.byteLength);
```

缓冲区用途定义 `usage`

usage 的属性值其他属性值参考文档:https://www.w3.org/TR/webgpu/#typedefdef-gpubufferusageflags

入门案例顶点缓冲区可以像下面一样设置，usage以后还会遇到其他的写法，遇到在专门讲解。

设置 `usage` 属性的值为 `GPUBufferUsage.VERTEX | GPUBufferUsage.COPY_DST` ，`|` 是JavaScript位运算符 。

`GPUBufferUsage.VERTEX` 表示用于该缓冲区是顶点缓冲区，就是存储顶点数据的缓冲区。

`GPUBufferUsage.COPY_DST` 的 `COPY` 是复制英文单词，DST是目的地单词destination的缩写，简单说该缓冲区可以写入顶点数据，作为复制顶点数据的目的地。

```js
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(作为顶点缓冲区|可以写入顶点数据)
    usage: GPUBufferUsage.VERTEX | GPUBufferUsage.COPY_DST,
});
```

顶点数据写入顶点缓冲区

GPU设备对象`device`队列属性`.queue`的有一个方法.writeBuffer()，功能是吧类型化数组中的数据写入`.createBuffer()`创建的顶点缓冲区中。

`.writeBuffer(vertexBuffer, 0, vertexArray)`表示把vertexArray里面的顶点数据写入到vertexBuffer对应的GPU显存缓冲区中，参数2表示从vertexArray获取顶点数据的偏移量(单位字节)，0表示从vertexArray的数据开头读取数据。

```js
//把vertexArray里面的顶点数据写入到vertexBuffer对应的GPU显存缓冲区中
//参数2的0表示从vertexArray的数据开头读取数据。
device.queue.writeBuffer(vertexBuffer, 0, vertexArray)
```

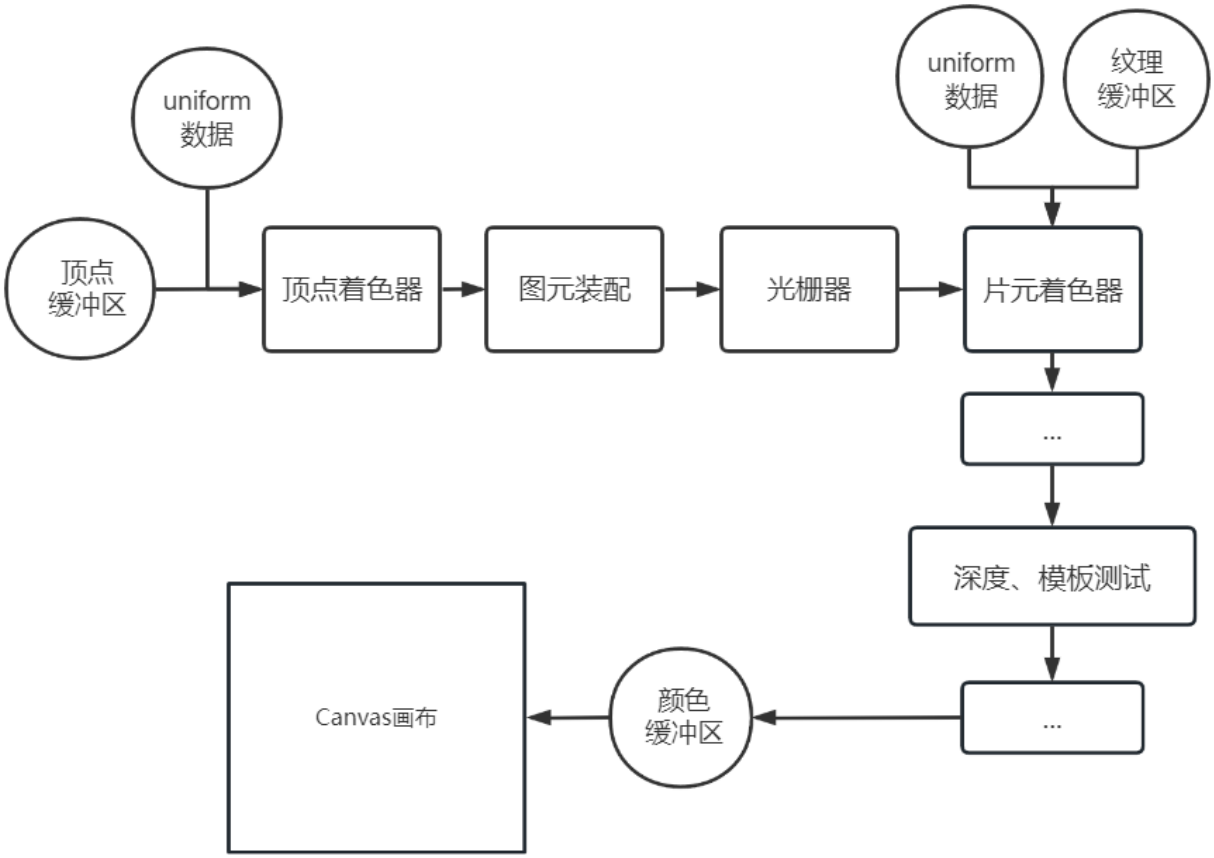
`.createRenderPipeline()`创建渲染管线

通过GPU设备对象的方法`.createRenderPipeline()`可以创建一个WebGPU渲染管线。

```js
// 创建一个WebGPU渲染管线对象pipeline
const pipeline = device.createRenderPipeline();
```

渲染管线你可以类比生活中的工厂流水线来理解，流水线上不同的功能单元，完成不同的零部件生产，对WebGPU渲染管线类似，WebGPU渲染管线上也提供用于3D渲染的不同功能单元，后面会一一讲解。

你可以把显卡比作一个工厂，工厂里面，你可以开设流水线，同样的道理，你也可以在显卡GPU上开设创建渲染管线，借助GPU设备对象的方法`.createRenderPipeline()`即可创建WebGPU的渲染管线，你可以根据需要创建多个渲染管线，当然咱们课程入门部分，只需要创建一个用来学习即可。



`.createRenderPipeline()` 参数

渲染管线方法 `.createRenderPipeline()` 的参数是一个对象，对象具有 `layout` 、`vertex` 、 `fragment` 、 `primitive` 等属性，这些属性对应了渲染管线上的不同功能单元。这些属性你现在还不理解，也没有关系，后面会给打逐步讲解。

```js
const pipeline = device.createRenderPipeline({
    layout: 'auto',
    vertex: {
        // 顶点着色器
        module: device.createShaderModule({ code: vertex }),
        entryPoint: "main"
    },
    fragment: {
        // 片元着色器
        module: device.createShaderModule({ code: fragment }),
        entryPoint: "main",
    },
    primitive: {
        topology: "triangle-list",//三角形绘制顶点数据
    }
```

```
});
```

`vertex.buffers` 配置顶点缓冲区

顶点缓冲区负责渲染管线提供顶点数据，所以所以需要通过渲染管线参数的 `vertex.buffers` 属性配置，渲染管线如何获取顶点缓冲区中的顶点数据。

```js
const pipeline = device.createRenderPipeline({
    vertex: {//顶点相关配置
        buffers: [// 顶点所有的缓冲区模块设置
            {//其中一个顶点缓冲区设置
                arrayStride: 3*4,//一个顶点数据占用的字节长度，该缓冲区一个顶点包含xyz
                attributes: [{// 顶点缓冲区属性
                    shaderLocation: 0,//GPU显存上顶点缓冲区标记存储位置
                    format: "float32x3",//格式：loat32x3表示一个顶点数据包含3个32位
                    offset: 0//arrayStride表示每组顶点数据间隔字节数，offset表示读取
                }]
            }
        ]
    },
});
```

← 2. WebGPU API和Canvas画布

4. 着色器语言WGSL快速了解→