升级Threejs 到161版本 & 迁移到WebGPU 可行性评估

结论

1. 是否升级：160，161这两个版本，新增了多范围更新position的方法，后面可以用到。建议升级。但是性能上开销更大了（大了一点点），CPU峰值更高，4D项目性能是瓶颈，而且这种多范围更新geometry的方法我们项目代码里暂时用不上，稳妥起见，建议不升级。
2. 迁移到WebGPU：不迁移。等到 Three.js 支持用 WGSL 语言编写 shader 的时候，再迁移。

原因

Three.js 整个项目中没有对 WGSL 的支持。那么 WebGPU demo 里对于 shader 的处理：见 https://github.com/mrdoob/three.js/blob/master/examples/webgpu_shadertoy.html 里面

```
class ShaderToyNode extends Nodes.Node {

                                constructor() {

                                        super( 'vec4' );

                                        this.mainImage = null;

                                }

                                transpile( glsl, iife = false ) {

                                        const decoder = new ShaderToyDecoder();

                                        const encoder = new TSLEncoder();
                                        encoder.iife = iife;
                                        encoder.uniqueNames = true;

                                        const jsCode = new Transpiler( decoder, encoder ).parse( glsl );

                                        return jsCode;

                                }
```

```
24
25                                  parse( glsl ) {
26
27                                      const jsCode = this.transpile( glsl,
    true );
28
29                                      const { mainImage } = eval( jsCode )(
    Nodes );
30
31                                      this.mainImage = mainImage;
32
33                                  }
34  }
```

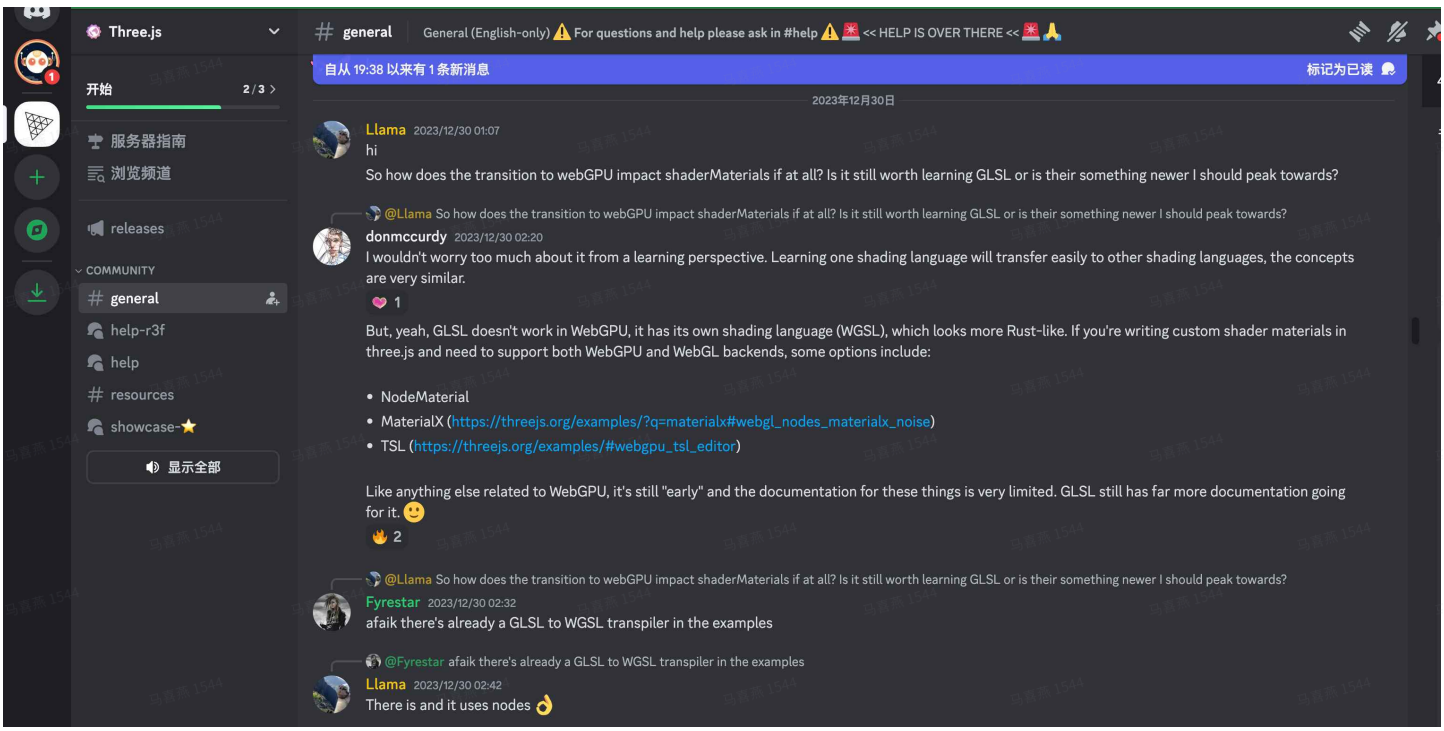
它的做法是把 GLSL 翻译成了 js， 用 js 处理节点，拿到处理过后的图片渲染函数，调用setup()方法，执行图片渲染函数，图片就显示出来了。

~~这是一种简易的替代方案，没有兼容性。无法像 shader 在各个着色阶段（顶点着色器、片元着色器、几何着色器等等）执行代码处理传进来的数据。~~

这是一种通过变换节点构造材质的方式（nodeMaterial）。我们项目用的是shaderMaterial（直接用GLSL编写，灵活性高）。

通用的方案是：熟悉 WebGPU 接口，熟悉 WebGL 接口，实现一套在 shader 各个阶段处理数据的语言，并改造渲染器，支持渲染处理过后的数据。（也叫 WebGPU shading language，WGSL。

或者等 Three.js 支持用 WGSL 编写材质。什么时候会支持？感觉要很久之后了。见 Three.js 仓库 discord 里面开发者的回复：



I wouldn't worry too much about it from a learning perspective. Learning one shading language will transfer easily to other shading languages, the concepts are very similar.

But, yeah, GLSL doesn't work in WebGPU, it has its own shading language (WGSL), which looks more Rust-like. If you're writing custom shader materials in three.js and need to support both WebGPU and WebGL backends, some options include:

- NodeMaterial
- MaterialX (https://threejs.org/examples/?q=materialx#webgl_nodes_materialx_noise)
- TSL (https://threejs.org/examples/#webgpu_tsl_editor)

Like anything else related to WebGPU, it's still "early" and the documentation for these things is very limited. GLSL still has far more documentation going for it.

上文给的三个选项：前两个是使用nodeMaterial，最后一个是使用 TSL（Three.js Shader Language），是使用一种更接近JavaScript的语法来编写着色器，支持webGPU；但它只是个抽象层，最终生成的着色器代码也是基于GLSL或WGSL的。综上：WGSL的支持还处在非常早期的阶段。

支持之后的迁移过程

1. 在 chrome://flags/ 里打开浏览器设置：WebGPU Developer Features 改成 Enabled。
2. 在 Stage.ts 文件里，用 WebGPURenderer 替换 WebGLRenderer。

```
import WebGPURenderer from
'three/examples/jsm/renderers/webgpu/WebGPURenderer';
```

3. 兼容所有用GLSL编写shader的地方。具体来说：
   a. GLSL翻译成WGSL。
   b. shaderMaterial也替换成WebGPU的shader。