TLS_PostProcessing

Looking at your code, the issue is likely that you're trying to update a TSL texture node's value after the material has been compiled, but the texture references in TSL are resolved during compilation time, not runtime.

Here are a few approaches that should work better:
Approach 1: Use a Uniform Node (Recommended)
Instead of using texture() directly, create a uniform that holds the texture reference:

```
javascriptconst postProcessing = new PostProcessing(renderer)

const scene1pass = pass(scene1, scene1.camera)
const scene2pass = pass(scene2, scene2.camera)

// Create a uniform node that can be updated at runtime
const activeSceneUniform = uniform(scene1pass.getTextureNode())

// Use the uniform in your post-processing chain
const outputPass = renderOutput(activeSceneUniform)

postProcessing.outputNode = outputPass

// To switch scenes:

activeSceneUniform.value = scene2pass.getTextureNode()

// Make sure to trigger a render after updating
```

Approach 2: Use textureLoad with Dynamic Reference

```
javascriptconst postProcessing = new PostProcessing(renderer)

const scene1pass = pass(scene1, scene1.camera)
const scene2pass = pass(scene2, scene2.camera)

// Create a uniform that holds the actual WebGL texture

const activeTextureUniform = uniform(scene1pass.getTexture('output'))

// Use textureLoad to sample from the uniform texture

const activeSceneTexture = textureLoad(activeTextureUniform,
screenUV)

const outputPass = renderOutput(activeSceneTexture)

postProcessing.outputNode = outputPass

// To switch scenes:

activeTextureUniform.value = scene2pass.getTexture('output')
```
Approach 3: Conditional Node (If you have a limited number of scenes)
```
javascriptconst postProcessing = new PostProcessing(renderer)

const scene1pass = pass(scene1, scene1.camera)
const scene2pass = pass(scene2, scene2.camera)

// Create a uniform to control which scene to show

const sceneSelector = uniform(0) // 0 for scene1, 1 for scene2

// Use conditional logic in the node graph

const activeSceneTexture = cond(
   sceneSelector.equal(0),
   texture(scene1pass.getTextureNode()),
   texture(scene2pass.getTextureNode())
)

const outputPass = renderOutput(activeSceneTexture)

postProcessing.outputNode = outputPass

// To switch scenes:

sceneSelector.value = 1 // Switch to scene2
```
Why Your Original Code Didn't Work

The issue with your original approach is that texture(scene1pass.getTextureNode()) creates a node that's bound to a specific texture during compilation. When you later try to update activeSceneTexture.value, you're trying to change the texture reference after the shader has already been compiled with the original texture.

Additional Notes

Make sure both scene passes are being rendered/updated each frame, even if only one is being displayed

You might need to call postProcessing.render() or trigger a render after updating the uniform

If you're still seeing issues, ensure that the texture formats and sizes match between your different scene passes

The first approach with uniform() should be the most straightforward and flexible solution for your use case.