**HTML**

<script id="2d-vertex-shader" type="x-shader/x-vertex">

varying vec2 textureCoordinates;

void main() {

textureCoordinates = uv;

gl\_Position = projectionMatrix \* modelViewMatrix \* vec4(position, 1.0);

}

</script>

<script id="2d-fragment-shader" type="x-shader/x-fragment">

varying vec2 textureCoordinates;

uniform sampler2D texture;

void main() {

vec4 textureFragment = texture2D(texture, textureCoordinates);

gl\_FragColor = textureFragment;

}

</script>

<div id="canvas"></div>

**CSS**

body {

align-items: center;

background-color: #edf0f6;

background-image: url('data:image/svg+xml,%3Csvg xmlns="http://www.w3.org/2000/svg" width="4" height="4" viewBox="0 0 4 4"%3E%3Cpath fill="%239C92AC" fill-opacity="0.4" d="M1 3h1v1H1V3zm2-2h1v1H3V1z"%3E%3C/path%3E%3C/svg%3E');

display: flex;

justify-content: center;

margin: 0;

min-height: 100vh;

}

#canvas {

height: 512px;

width: 512px;

}

**JS**

let canvas = document.getElementById('canvas');

let image = 'https://source.unsplash.com/DGQhTvIBfV4/1024x1024';

let fragmentShader = document.getElementById('2d-fragment-shader').innerHTML;

let vertexShader = document.getElementById('2d-vertex-shader').innerHTML;

let scene = new THREE.Scene();

let camera = new THREE.OrthographicCamera(

canvas.offsetWidth / -2.0,

canvas.offsetWidth / 2.0,

canvas.offsetHeight / 2.0,

canvas.offsetHeight / -2.0,

1.0,

1000

);

camera.position.z = 1.0;

let renderer = new THREE.WebGLRenderer({ antialias: false });

renderer.setPixelRatio(window.devicePixelRatio);

renderer.setClearColor(0xffffff, 0.0);

renderer.setSize(canvas.offsetWidth, canvas.offsetHeight);

canvas.appendChild(renderer.domElement);

let loader = new THREE.TextureLoader();

loader.setCrossOrigin('');

let render = () => {

renderer.render(scene, camera);

};

let texture = loader.load(image, render);

texture.minFilter = THREE.LinearFilter; // LinearMipMapLinearFilter

let material = new THREE.ShaderMaterial({

fragmentShader: fragmentShader,

uniforms: {

texture: { type: 't', value: texture },

time: { value: 1.0 },

},

vertexShader: vertexShader

});

let geometry = new THREE.PlaneBufferGeometry(

canvas.offsetWidth,

canvas.offsetHeight,

1.0

);

let mesh = new THREE.Mesh(geometry, material);

scene.add(mesh);

let time = 0;

let animate = () => {

time += 0.01;

material.uniforms.time.value += time;

material.needsUpdate = true;

renderer.render(scene, camera);

requestAnimationFrame(() => {

animate();

});

};

// animate();