import \* as THREE from "https://cdn.skypack.dev/three@0.136.0";

import {OrbitControls} from "https://cdn.skypack.dev/three@0.136.0/examples/jsm/controls/OrbitControls";

console.clear();

let scene = new THREE.Scene();

scene.background = new THREE.Color(0x160016);

let camera = new THREE.PerspectiveCamera(60, innerWidth / innerHeight, 1, 1000);

camera.position.set(0, 4, 21);

let renderer = new THREE.WebGLRenderer();

renderer.setSize(innerWidth, innerHeight);

document.body.appendChild(renderer.domElement);

window.addEventListener("resize", event => {

camera.aspect = innerWidth / innerHeight;

camera.updateProjectionMatrix();

renderer.setSize(innerWidth, innerHeight);

})

let controls = new OrbitControls(camera, renderer.domElement);

controls.enableDamping = true;

controls.enablePan = false;

let gu = {

time: {value: 0}

}

let sizes = [];

let shift = [];

let pushShift = () => {

shift.push(

Math.random() \* Math.PI,

Math.random() \* Math.PI \* 2,

(Math.random() \* 0.9 + 0.1) \* Math.PI \* 0.1,

Math.random() \* 0.9 + 0.1

);

}

let pts = new Array(25000).fill().map(p => {

sizes.push(Math.random() \* 1.5 + 0.5);

pushShift();

return new THREE.Vector3().randomDirection().multiplyScalar(Math.random() \* 0.5 + 9.5);

})

for(let i = 0; i < 50000; i++){

let r = 10, R = 40;

let rand = Math.pow(Math.random(), 1.5);

let radius = Math.sqrt(R \* R \* rand + (1 - rand) \* r \* r);

pts.push(new THREE.Vector3().setFromCylindricalCoords(radius, Math.random() \* 2 \* Math.PI, (Math.random() - 0.5) \* 2 ));

sizes.push(Math.random() \* 1.5 + 0.5);

pushShift();

}

let g = new THREE.BufferGeometry().setFromPoints(pts);

g.setAttribute("sizes", new THREE.Float32BufferAttribute(sizes, 1));

g.setAttribute("shift", new THREE.Float32BufferAttribute(shift, 4));

let m = new THREE.PointsMaterial({

size: 0.1,

transparent: true,

blending: THREE.AdditiveBlending,

onBeforeCompile: shader => {

shader.uniforms.time = gu.time;

shader.vertexShader = `

uniform float time;

attribute float sizes;

attribute vec4 shift;

varying vec3 vColor;

${shader.vertexShader}

`.replace(

`gl\_PointSize = size;`,

`gl\_PointSize = size \* sizes;`

).replace(

`#include <color\_vertex>`,

`#include <color\_vertex>

float d = length(abs(position) / vec3(40., 10., 40));

d = clamp(d, 0., 1.);

vColor = mix(vec3(227., 155., 0.), vec3(100., 50., 255.), d) / 255.;

`

).replace(

`#include <begin\_vertex>`,

`#include <begin\_vertex>

float t = time;

float moveT = mod(shift.x + shift.z \* t, PI2);

float moveS = mod(shift.y + shift.z \* t, PI2);

transformed += vec3(cos(moveS) \* sin(moveT), cos(moveT), sin(moveS) \* sin(moveT)) \* shift.a;

`

);

console.log(shader.vertexShader);

shader.fragmentShader = `

varying vec3 vColor;

${shader.fragmentShader}

`.replace(

`#include <clipping\_planes\_fragment>`,

`#include <clipping\_planes\_fragment>

float d = length(gl\_PointCoord.xy - 0.5);

if (d > 0.5) discard;

`

).replace(

`vec4 diffuseColor = vec4( diffuse, opacity );`,

`vec4 diffuseColor = vec4( vColor, smoothstep(0.5, 0.2, d) \* 0.5 + 0.5 );`

);

console.log(shader.fragmentShader);

}

});

let p = new THREE.Points(g, m);

p.rotation.order = "ZYX";

p.rotation.z = 0.2;

scene.add(p)

let clock = new THREE.Clock();

renderer.setAnimationLoop(() => {

controls.update();

let t = clock.getElapsedTime() \* 0.5;

gu.time.value = t \* Math.PI;

p.rotation.y = t \* 0.05;

renderer.render(scene, camera);

});