<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Black Hole Visualization</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div id="blackhole">

<div class="centerHover"><span>ENTER</span></div>

</div>

<script src="script.js"></script>

</body>

</html>

body, html {

height: 100%;

margin: 0;

padding: 0;

}

body {

height: 100%;

background-color: rgba(25,25,25,1);

overflow: hidden;

}

#blackhole {

height: 100%;

width: 100%;

position: relative;

display: flex;

}

.centerHover {

width: 255px;

height: 255px;

background-color: transparent;

border-radius: 50%;

position: absolute;

left: 50%;

top: 50%;

margin-top: -128px;

margin-left: -128px;

z-index: 2;

cursor: pointer;

line-height: 255px;

text-align: center;

transition: all 500ms;

}

.centerHover.open {

opacity: 0;

pointer-events: none;

}

.centerHover:hover span {

color: #DDD;

}

.centerHover:hover span:before {

background-color: #DDD;

}

.centerHover:hover span:after {

background-color: #DDD;

}

.centerHover span {

color: #666;

font-family: serif;

font-size: 18px;

position: relative;

transition: all 500ms;

}

.centerHover span:before {

content: '';

display: inline-block;

height: 1px;

width: 16px;

margin-right: 12px;

margin-bottom: 4px;

background-color: #666;

transition: all 500ms;

}

.centerHover span:after {

content: '';

display: inline-block;

height: 1px;

width: 16px;

margin-left: 12px;

margin-bottom: 4px;

background-color: #666;

transition: all 500ms;

}

canvas {

position: relative;

z-index: 1;

width: 100%;

height: 100%;

margin: auto;

}

#blackhole {

height: 100%;

width: 100%;

position: relative;

display: flex;

}

.centerHover {

width: 255px;

height: 255px;

background-color: transparent;

border-radius: 50%;

position: absolute;

left: 50%;

top: 50%;

margin-top: -128px;

margin-left: -128px;

z-index: 2;

cursor: pointer;

line-height: 255px;

text-align: center;

transition: all 500ms;

}

.centerHover.open {

opacity: 0;

pointer-events: none;

}

.centerHover:hover span {

color: #DDD;

}

.centerHover:hover span:before {

background-color: #DDD;

}

.centerHover:hover span:after {

background-color: #DDD;

}

.centerHover span {

color: #666;

font-family: serif;

font-size: 18px;

position: relative;

transition: all 500ms;

}

.centerHover span:before {

content: '';

display: inline-block;

height: 1px;

width: 16px;

margin-right: 12px;

margin-bottom: 4px;

background-color: #666;

transition: all 500ms;

}

.centerHover span:after {

content: '';

display: inline-block;

height: 1px;

width: 16px;

margin-left: 12px;

margin-bottom: 4px;

background-color: #666;

transition: all 500ms;

}

canvas {

position: relative;

z-index: 1;

width: 100%;

height: 100%;

margin: auto;

}

function blackhole(element) {

const container = document.querySelector(element);

const h = container.offsetHeight;

const w = container.offsetWidth;

const cw = w;

const ch = h;

const maxorbit = 255; // distance from center

const centery = ch / 2;

const centerx = cw / 2;

const startTime = new Date().getTime();

let currentTime = 0;

const stars = [];

let collapse = false; // if hovered

let expanse = false; // if clicked

let returning = false; // if particles are returning to orbit

// Create canvas

const canvas = document.createElement('canvas');

canvas.width = cw;

canvas.height = ch;

container.appendChild(canvas);

const context = canvas.getContext("2d");

context.globalCompositeOperation = "multiply";

function setDPI(canvas, dpi) {

// Set up CSS size if it's not set up already

if (!canvas.style.width)

canvas.style.width = canvas.width + 'px';

if (!canvas.style.height)

canvas.style.height = canvas.height + 'px';

const scaleFactor = dpi / 96;

canvas.width = Math.ceil(canvas.width \* scaleFactor);

canvas.height = Math.ceil(canvas.height \* scaleFactor);

const ctx = canvas.getContext('2d');

ctx.scale(scaleFactor, scaleFactor);

}

function rotate(cx, cy, x, y, angle) {

const radians = angle;

const cos = Math.cos(radians);

const sin = Math.sin(radians);

const nx = (cos \* (x - cx)) + (sin \* (y - cy)) + cx;

const ny = (cos \* (y - cy)) - (sin \* (x - cx)) + cy;

return [nx, ny];

}

setDPI(canvas, 192);

class Star {

constructor() {

// Get a weighted random number, so that the majority of stars will form in the center of the orbit

const rands = [];

rands.push(Math.random() \* (maxorbit / 2) + 1);

rands.push(Math.random() \* (maxorbit / 2) + maxorbit);

this.orbital = (rands.reduce((p, c) => p + c, 0) / rands.length);

this.x = centerx; // All of these stars are at the center x position at all times

this.y = centery + this.orbital; // Set Y position starting at the center y + the position in the orbit

this.yOrigin = centery + this.orbital; // this is used to track the particles origin

this.speed = (Math.floor(Math.random() \* 2.5) + 1.5) \* Math.PI / 180; // The rate at which this star will orbit

this.rotation = 0; // current Rotation

this.startRotation = (Math.floor(Math.random() \* 360) + 1) \* Math.PI / 180; // Starting rotation

this.id = stars.length; // This will be used when expansion takes place

this.collapseBonus = this.orbital - (maxorbit \* 0.7); // This "bonus" is used to randomly place some stars outside of the blackhole on hover

if (this.collapseBonus < 0) { // if the collapse "bonus" is negative

this.collapseBonus = 0; // set it to 0, this way no stars will go inside the blackhole

}

this.color = 'rgba(255,255,255,' + (1 - ((this.orbital) / 255)) + ')'; // Color the star white, but make it more transparent the further out it is generated

this.hoverPos = centery + (maxorbit / 2) + this.collapseBonus; // Where the star will go on hover of the blackhole

this.expansePos = centery + (this.id % 100) \* -10 + (Math.floor(Math.random() \* 20) + 1); // Where the star will go when expansion takes place

this.prevR = this.startRotation;

this.prevX = this.x;

this.prevY = this.y;

// Store original position for returning

this.originalY = this.yOrigin;

stars.push(this);

}

draw() {

if (!expanse && !returning) {

this.rotation = this.startRotation + (currentTime \* this.speed);

if (!collapse) { // not hovered

if (this.y > this.yOrigin) {

this.y -= 2.5;

}

if (this.y < this.yOrigin - 4) {

this.y += (this.yOrigin - this.y) / 10;

}

} else { // on hover

this.trail = 1;

if (this.y > this.hoverPos) {

this.y -= (this.hoverPos - this.y) / -5;

}

if (this.y < this.hoverPos - 4) {

this.y += 2.5;

}

}

} else if (expanse && !returning) {

this.rotation = this.startRotation + (currentTime \* (this.speed / 2));

if (this.y > this.expansePos) {

this.y -= Math.floor(this.expansePos - this.y) / -80; // Slower expansion for better visibility

}

} else if (returning) {

// Returning to original orbit slowly

this.rotation = this.startRotation + (currentTime \* this.speed);

if (Math.abs(this.y - this.originalY) > 2) {

this.y += (this.originalY - this.y) / 50; // Much slower return

} else {

this.y = this.originalY;

this.yOrigin = this.originalY;

}

}

context.save();

context.fillStyle = this.color;

context.strokeStyle = this.color;

context.beginPath();

const oldPos = rotate(centerx, centery, this.prevX, this.prevY, -this.prevR);

context.moveTo(oldPos[0], oldPos[1]);

context.translate(centerx, centery);

context.rotate(this.rotation);

context.translate(-centerx, -centery);

context.lineTo(this.x, this.y);

context.stroke();

context.restore();

this.prevR = this.rotation;

this.prevX = this.x;

this.prevY = this.y;

}

}

// Event listeners

const centerHover = document.querySelector('.centerHover');

centerHover.addEventListener('click', function() {

collapse = false;

expanse = true;

returning = false;

this.classList.add('open');

// Start the return cycle after full expansion (20-30 seconds)

setTimeout(() => {

expanse = false;

returning = true;

// After particles return, reset to normal orbit

setTimeout(() => {

returning = false;

this.classList.remove('open');

}, 8000); // 8 seconds to return slowly

}, 25000); // 25 seconds of expansion experience

});

centerHover.addEventListener('mouseover', function() {

if (expanse === false) {

collapse = true;

}

});

centerHover.addEventListener('mouseout', function() {

if (expanse === false) {

collapse = false;

}

});

// Animation loop

function loop() {

const now = new Date().getTime();

currentTime = (now - startTime) / 50;

context.fillStyle = 'rgba(25,25,25,0.2)'; // somewhat clear the context, this way there will be trails behind the stars

context.fillRect(0, 0, cw, ch);

for (let i = 0; i < stars.length; i++) { // For each star

if (stars[i] !== undefined) {

stars[i].draw(); // Draw it

}

}

requestAnimationFrame(loop);

}

function init() {

context.fillStyle = 'rgba(25,25,25,1)'; // Initial clear of the canvas

context.fillRect(0, 0, cw, ch);

for (let i = 0; i < 2500; i++) { // create 2500 stars

new Star();

}

loop();

}

init();

}

// Initialize when DOM is loaded

document.addEventListener('DOMContentLoaded', () => {

blackhole('#blackhole');

});