mở trang : <https://jsfiddle.net>

ô html điền

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

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

<title>Document</title>

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

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

</head>

<body>

<canvas id="pinkboard"></canvas>

<div id="drag-container">

<div id="spin-container">

<img

src="https://drive.google.com/uc?id=1r850UYdj3JSCHcbwwER4Byhu3XuL387S"

alt=""

/>

<img

src="https://drive.google.com/uc?id=1r850UYdj3JSCHcbwwER4Byhu3XuL387S"

alt=""

/>

<img

src="https://drive.google.com/uc?id=1r850UYdj3JSCHcbwwER4Byhu3XuL387S"

alt=""

/>

<img

src="https://drive.google.com/uc?id=1r850UYdj3JSCHcbwwER4Byhu3XuL387S"

alt=""

/>

<img

src="https://drive.google.com/uc?id=1r850UYdj3JSCHcbwwER4Byhu3XuL387S"

alt=""

/>

<img

src="https://drive.google.com/uc?id=1r850UYdj3JSCHcbwwER4Byhu3XuL387S"

alt=""

/>

<!-- Example add video -->

</div>

<div id="ground"></div>

</div>

<div id="music-container">

<iframe

src="https://www.youtube.com/embed/yrAlRngYndg?autoplay=1"

class="frame"

allowfullscreen

frameborder="0"

width="420"

height="345"

allow="autoplay"

></iframe>

</div>

</body>

<script>

// You can change global variables here:

var radius = 240; // how big of the radius

var autoRotate = true; // auto rotate or not

var rotateSpeed = -60; // unit: seconds/360 degrees

var imgWidth = 120; // width of images (unit: px)

var imgHeight = 170; // height of images (unit: px)

// Link of background music - set 'null' if you dont want to play background music

var bgMusicURL =

"https://api-v2.soundcloud.com/tracks/308486718/albums?representation=mini&client\_id=jOJjarVXJfZlI309Up55k93EUDG7ILW6&limit=10&offset=0&linked\_partitioning=1&app\_version=1666004844&app\_locale=en";

var bgMusicControls = false; // Show UI music control

var settings = {

particles: {

length: 500, // maximum amount of particles

duration: 2, // particle duration in sec

velocity: 100, // particle velocity in pixels/sec

effect: -0.75, // play with this for a nice effect

size: 30, // particle size in pixels

},

};

/\*

\* RequestAnimationFrame polyfill by Erik Möller

\*/

(function () {

var b = 0;

var c = ["ms", "moz", "webkit", "o"];

for (var a = 0; a < c.length && !window.requestAnimationFrame; ++a) {

window.requestAnimationFrame = window[c[a] + "RequestAnimationFrame"];

window.cancelAnimationFrame =

window[c[a] + "CancelAnimationFrame"] ||

window[c[a] + "CancelRequestAnimationFrame"];

}

if (!window.requestAnimationFrame) {

window.requestAnimationFrame = function (h, e) {

var d = new Date().getTime();

var f = Math.max(0, 16 - (d - b));

var g = window.setTimeout(function () {

h(d + f);

}, f);

b = d + f;

return g;

};

}

if (!window.cancelAnimationFrame) {

window.cancelAnimationFrame = function (d) {

clearTimeout(d);

};

}

})();

/\*

\* Point class

\*/

var Point = (function () {

function Point(x, y) {

this.x = typeof x !== "undefined" ? x : 0;

this.y = typeof y !== "undefined" ? y : 0;

}

Point.prototype.clone = function () {

return new Point(this.x, this.y);

};

Point.prototype.length = function (length) {

if (typeof length == "undefined")

return Math.sqrt(this.x \* this.x + this.y \* this.y);

this.normalize();

this.x \*= length;

this.y \*= length;

return this;

};

Point.prototype.normalize = function () {

var length = this.length();

this.x /= length;

this.y /= length;

return this;

};

return Point;

})();

/\*

\* Particle class

\*/

var Particle = (function () {

function Particle() {

this.position = new Point();

this.velocity = new Point();

this.acceleration = new Point();

this.age = 0;

}

Particle.prototype.initialize = function (x, y, dx, dy) {

this.position.x = x;

this.position.y = y;

this.velocity.x = dx;

this.velocity.y = dy;

this.acceleration.x = dx \* settings.particles.effect;

this.acceleration.y = dy \* settings.particles.effect;

this.age = 0;

};

Particle.prototype.update = function (deltaTime) {

this.position.x += this.velocity.x \* deltaTime;

this.position.y += this.velocity.y \* deltaTime;

this.velocity.x += this.acceleration.x \* deltaTime;

this.velocity.y += this.acceleration.y \* deltaTime;

this.age += deltaTime;

};

Particle.prototype.draw = function (context, image) {

function ease(t) {

return --t \* t \* t + 1;

}

var size = image.width \* ease(this.age / settings.particles.duration);

context.globalAlpha = 1 - this.age / settings.particles.duration;

context.drawImage(

image,

this.position.x - size / 2,

this.position.y - size / 2,

size,

size

);

};

return Particle;

})();

/\*

\* ParticlePool class

\*/

var ParticlePool = (function () {

var particles,

firstActive = 0,

firstFree = 0,

duration = settings.particles.duration;

function ParticlePool(length) {

// create and populate particle pool

particles = new Array(length);

for (var i = 0; i < particles.length; i++)

particles[i] = new Particle();

}

ParticlePool.prototype.add = function (x, y, dx, dy) {

particles[firstFree].initialize(x, y, dx, dy);

// handle circular queue

firstFree++;

if (firstFree == particles.length) firstFree = 0;

if (firstActive == firstFree) firstActive++;

if (firstActive == particles.length) firstActive = 0;

};

ParticlePool.prototype.update = function (deltaTime) {

var i;

// update active particles

if (firstActive < firstFree) {

for (i = firstActive; i < firstFree; i++)

particles[i].update(deltaTime);

}

if (firstFree < firstActive) {

for (i = firstActive; i < particles.length; i++)

particles[i].update(deltaTime);

for (i = 0; i < firstFree; i++) particles[i].update(deltaTime);

}

// remove inactive particles

while (

particles[firstActive].age >= duration &&

firstActive != firstFree

) {

firstActive++;

if (firstActive == particles.length) firstActive = 0;

}

};

ParticlePool.prototype.draw = function (context, image) {

// draw active particles

if (firstActive < firstFree) {

for (i = firstActive; i < firstFree; i++)

particles[i].draw(context, image);

}

if (firstFree < firstActive) {

for (i = firstActive; i < particles.length; i++)

particles[i].draw(context, image);

for (i = 0; i < firstFree; i++) particles[i].draw(context, image);

}

};

return ParticlePool;

})();

/\*

\* Putting it all together

\*/

(function (canvas) {

var context = canvas.getContext("2d"),

particles = new ParticlePool(settings.particles.length),

particleRate = settings.particles.length / settings.particles.duration, // particles/sec

time;

// get point on heart with -PI <= t <= PI

function pointOnHeart(t) {

return new Point(

160 \* Math.pow(Math.sin(t), 3),

130 \* Math.cos(t) -

50 \* Math.cos(2 \* t) -

20 \* Math.cos(3 \* t) -

10 \* Math.cos(4 \* t) +

25

);

}

// creating the particle image using a dummy canvas

var image = (function () {

var canvas = document.createElement("canvas"),

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

canvas.width = settings.particles.size;

canvas.height = settings.particles.size;

// helper function to create the path

function to(t) {

var point = pointOnHeart(t);

point.x =

settings.particles.size / 2 +

(point.x \* settings.particles.size) / 350;

point.y =

settings.particles.size / 2 -

(point.y \* settings.particles.size) / 350;

return point;

}

// create the path

context.beginPath();

var t = -Math.PI;

var point = to(t);

context.moveTo(point.x, point.y);

while (t < Math.PI) {

t += 0.01; // baby steps!

point = to(t);

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

}

context.closePath();

// create the fill

context.fillStyle = "#ea80b0";

context.fill();

// create the image

var image = new Image();

image.src = canvas.toDataURL();

return image;

})();

// render that thing!

function render() {

// next animation frame

requestAnimationFrame(render);

// update time

var newTime = new Date().getTime() / 1000,

deltaTime = newTime - (time || newTime);

time = newTime;

// clear canvas

context.clearRect(0, 0, canvas.width, canvas.height);

// create new particles

var amount = particleRate \* deltaTime;

for (var i = 0; i < amount; i++) {

var pos = pointOnHeart(Math.PI - 2 \* Math.PI \* Math.random());

var dir = pos.clone().length(settings.particles.velocity);

particles.add(

canvas.width / 2 + pos.x,

canvas.height / 2 - pos.y,

dir.x,

-dir.y

);

}

// update and draw particles

particles.update(deltaTime);

particles.draw(context, image);

}

// handle (re-)sizing of the canvas

function onResize() {

canvas.width = canvas.clientWidth;

canvas.height = canvas.clientHeight;

}

window.onresize = onResize;

// delay rendering bootstrap

setTimeout(function () {

onResize();

render();

}, 10);

})(document.getElementById("pinkboard"));

// ===================== start =======================

// animation start after 1000 miliseconds

setTimeout(init, 1000);

var odrag = document.getElementById("drag-container");

var ospin = document.getElementById("spin-container");

var aImg = ospin.getElementsByTagName("img");

var aVid = ospin.getElementsByTagName("video");

var aEle = [...aImg, ...aVid]; // combine 2 arrays

// Size of images

ospin.style.width = imgWidth + "px";

ospin.style.height = imgHeight + "px";

// Size of ground - depend on radius

var ground = document.getElementById("ground");

ground.style.width = radius \* 3 + "px";

ground.style.height = radius \* 3 + "px";

function init(delayTime) {

for (var i = 0; i < aEle.length; i++) {

aEle[i].style.transform =

"rotateY(" +

i \* (360 / aEle.length) +

"deg) translateZ(" +

radius +

"px)";

aEle[i].style.transition = "transform 1s";

aEle[i].style.transitionDelay =

delayTime || (aEle.length - i) / 4 + "s";

}

}

function applyTranform(obj) {

// Constrain the angle of camera (between 0 and 180)

if (tY > 180) tY = 180;

if (tY < 0) tY = 0;

// Apply the angle

obj.style.transform = "rotateX(" + -tY + "deg) rotateY(" + tX + "deg)";

}

function playSpin(yes) {

ospin.style.animationPlayState = yes ? "running" : "paused";

}

var sX,

sY,

nX,

nY,

desX = 0,

desY = 0,

tX = 0,

tY = 10;

// auto spin

if (autoRotate) {

var animationName = rotateSpeed > 0 ? "spin" : "spinRevert";

ospin.style.animation = `${animationName} ${Math.abs(

rotateSpeed

)}s infinite linear`;

}

// add background music

if (bgMusicURL) {

document.getElementById("music-container").innerHTML += `

<audio src="${bgMusicURL}" ${

bgMusicControls ? "controls" : ""

} autoplay loop>

<p>If you are reading this, it is because your browser does not support the audio element.</p>

</audio>

`;

}

// setup events

document.onpointerdown = function (e) {

clearInterval(odrag.timer);

e = e || window.event;

var sX = e.clientX,

sY = e.clientY;

this.onpointermove = function (e) {

e = e || window.event;

var nX = e.clientX,

nY = e.clientY;

desX = nX - sX;

desY = nY - sY;

tX += desX \* 0.1;

tY += desY \* 0.1;

applyTranform(odrag);

sX = nX;

sY = nY;

};

this.onpointerup = function (e) {

odrag.timer = setInterval(function () {

desX \*= 0.95;

desY \*= 0.95;

tX += desX \* 0.1;

tY += desY \* 0.1;

applyTranform(odrag);

playSpin(false);

if (Math.abs(desX) < 0.5 && Math.abs(desY) < 0.5) {

clearInterval(odrag.timer);

playSpin(true);

}

}, 17);

this.onpointermove = this.onpointerup = null;

};

return false;

};

document.onmousewheel = function (e) {

e = e || window.event;

var d = e.wheelDelta / 20 || -e.detail;

radius += d;

init(1);

};

</script>

</html>

**Với ô css thfi điền :**

\* { margin: 0; padding: 0; } .frame { width: 30px; height: 30px; } canvas { position: absolute; width: 100%; height: 100%; } html, body { height: 100%; /\* for touch screen \*/ touch-action: none; } body { overflow: hidden; display: -webkit-box; display: -ms-flexbox; display: flex; background: #111; -webkit-perspective: 1000px; perspective: 1000px; -webkit-transform-style: preserve-3d; transform-style: preserve-3d; } #drag-container, #spin-container { position: relative; display: -webkit-box; display: -ms-flexbox; display: flex; margin: auto; -webkit-transform-style: preserve-3d; transform-style: preserve-3d; -webkit-transform: rotateX(-10deg); transform: rotateX(-10deg); } #drag-container img, #drag-container video { -webkit-transform-style: preserve-3d; transform-style: preserve-3d; position: absolute; left: 0; top: 0; width: 100%; height: 100%; line-height: 200px; font-size: 50px; text-align: center; -webkit-box-shadow: 0 0 8px #fff; box-shadow: 0 0 8px #fff; -webkit-box-reflect: below 10px linear-gradient(transparent, transparent, #0005); } #drag-container img:hover, #drag-container video:hover { -webkit-box-shadow: 0 0 15px #fffd; box-shadow: 0 0 15px #fffd; -webkit-box-reflect: below 10px linear-gradient(transparent, transparent, #0007); } #drag-container p { font-family: Serif; position: absolute; top: 100%; left: 50%; -webkit-transform: translate(-50%, -50%) rotateX(90deg); transform: translate(-50%, -50%) rotateX(90deg); color: #fff; } #ground { width: 900px; height: 900px; position: absolute; top: 100%; left: 50%; -webkit-transform: translate(-50%, -50%) rotateX(90deg); transform: translate(-50%, -50%) rotateX(90deg); background: -webkit-radial-gradient( center center, farthest-side, #9993, transparent ); } #music-container { position: absolute; top: 0; left: 0; } @-webkit-keyframes spin { from { -webkit-transform: rotateY(0deg); transform: rotateY(0deg); } to { -webkit-transform: rotateY(360deg); transform: rotateY(360deg); } } @keyframes spin { from { -webkit-transform: rotateY(0deg); transform: rotateY(0deg); } to { -webkit-transform: rotateY(360deg); transform: rotateY(360deg); } } @-webkit-keyframes spinRevert { from { -webkit-transform: rotateY(360deg); transform: rotateY(360deg); } to { -webkit-transform: rotateY(0deg); transform: rotateY(0deg); } } @keyframes spinRevert { from { -webkit-transform: rotateY(360deg); transform: rotateY(360deg); } to { -webkit-transform: rotateY(0deg); transform: rotateY(0deg); } } html, body { overflow: hidden; padding: 0; margin: 0; background: #000; } canvas { position: absolute; width: 100%; height: 100%; } canvas { /\* top: 50%; left: 50%; \*/ z-index: 1; display: block; position: absolute; transform: translate(-50%, -50%); animation: heart 1.5s ease infinite; } @keyframes heart { 0% { transform: scale(1); } 30% { transform: scale(0.8); } /\* 60% { transform: scale(1.2); } \*/ 100% { transform: scale(1); } }

Chỗ : src thì thay bằng ảnh muốn hiện