<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>爱心</title><!-- 这是网页标题 -->

<style>

body{

overflow: hidden;

margin: 0;

}

h1{

position: fixed;

top: 30%;

left: 0;

width: 100%;

text-align: center;

transform:translateY(-50%);

font-family: 'Love Ya Like A Sister', cursive;

font-size: 60px;

color: #c70012;

padding: 0 20px;

}

h1 span{

position: fixed;

left: 0;

width: 100%;

text-align: center;

margin-top:30px;

font-size:40px;

}

</style>

</head>

<body>

<h1 id="h1"></h1>

<canvas></canvas> <!--canvas 画布-->

<script>

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

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

var ww,wh;

function onResize(){

ww = canvas.width = window.innerWidth;

wh = canvas.height = window.innerHeight;

}

ctx.strokeStyle = "red";

ctx.shadowBlur = 25;

ctx.shadowColor = "hsla(0, 100%, 60%,0.5)";

var precision = 100;

var hearts = [];

var mouseMoved = false;

function onMove(e){

mouseMoved = true;

if(e.type === "touchmove"){

hearts.push(new Heart(e.touches[0].clientX, e.touches[0].clientY));

hearts.push(new Heart(e.touches[0].clientX, e.touches[0].clientY));

}

else{

hearts.push(new Heart(e.clientX, e.clientY));

hearts.push(new Heart(e.clientX, e.clientY));

}

}

var Heart = function(x,y){

this.x = x || Math.random()\*ww;

this.y = y || Math.random()\*wh;

this.size = Math.random()\*2 + 1;

this.shadowBlur = Math.random() \* 10;

this.speedX = (Math.random()+0.2-0.6) \* 8;

this.speedY = (Math.random()+0.2-0.6) \* 8;

this.speedSize = Math.random()\*0.05 + 0.01;

this.opacity = 1;

this.vertices = [];

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

var step = (i / precision - 0.5) \* (Math.PI \* 2);

var vector = {

x : (15 \* Math.pow(Math.sin(step), 3)),

y : -(13 \* Math.cos(step) - 5 \* Math.cos(2 \* step) - 2 \* Math.cos(3 \* step) - Math.cos(4 \* step))

}

this.vertices.push(vector);

}

}

Heart.prototype.draw = function(){

this.size -= this.speedSize;

this.x += this.speedX;

this.y += this.speedY;

ctx.save();

ctx.translate(-1000,this.y);

ctx.scale(this.size, this.size);

ctx.beginPath();

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

var vector = this.vertices[i];

ctx.lineTo(vector.x, vector.y);

}

ctx.globalAlpha = this.size;

ctx.shadowBlur = Math.round((3 - this.size) \* 10);

ctx.shadowColor = "hsla(0, 100%, 60%,0.5)";

ctx.shadowOffsetX = this.x + 1000;

ctx.globalCompositeOperation = "screen"

ctx.closePath();

ctx.fill()

ctx.restore();

};

function render(a){

requestAnimationFrame(render);

hearts.push(new Heart())

ctx.clearRect(0,0,ww,wh);

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

hearts[i].draw();

if(hearts[i].size <= 0){

hearts.splice(i,1);

i--;

}

}

}

onResize();

window.addEventListener("mousemove", onMove);

window.addEventListener("touchmove", onMove);

window.addEventListener("resize", onResize);

requestAnimationFrame(render);

window.οnlοad=function starttime(){

time(h1,'2022,2,1'); // 2021年春节时间

ptimer = setTimeout(starttime,1000); // 添加计时器

}

</script>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<title></title>

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

</head>

<style>

\* {

padding: 0;

margin: 0;

}

html,

body {

height: 100%;

padding: 0;

margin: 0;

background: #000;

}

.aa {

position: fixed;

left: 50%;

bottom: 10px;

color: #ccc;

}

.container {

width: 100%;

height: 100%;

}

canvas {

z-index: 99;

position: absolute;

width: 100%;

height: 100%;

}

#juzhong{

position: absolute;

left: 50%;

top: 50%;

}

</style>

<body>

<!-- 樱花 -->

<div id="jsi-cherry-container" class="container">

<div id="juzhong" >

<!-- 填写名字 -->

<font color="hotpink"></font>>

</div>

<audio autoplay="autopaly">

<source src="renxi.mp3" type="audio/mp3" />

</audio>

<img class="img" src="./123.png" alt="" />

<!-- 爱心 -->

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

</div>

</body>

</html>

<script>

/\*

\* Settings

\*/

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

},

};

(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"));

</script>

<script>

var RENDERER = {

INIT\_CHERRY\_BLOSSOM\_COUNT: 30,

MAX\_ADDING\_INTERVAL: 10,

init: function () {

this.setParameters();

this.reconstructMethods();

this.createCherries();

this.render();

if (

navigator.userAgent.match(

/(phone|pod|iPhone|iPod|ios|Android|Mobile|BlackBerry|IEMobile|MQQBrowser|JUC|Fennec|wOSBrowser|BrowserNG|WebOS|Symbian|Windows Phone)/i

)

) {

// var box = document.querySelectorAll(".box")[0];

// console.log(box, "移动端");

// box.style.marginTop = "65%";

}

},

setParameters: function () {

this.$container = $("#jsi-cherry-container");

this.width = this.$container.width();

this.height = this.$container.height();

this.context = $("<canvas />")

.attr({ width: this.width, height: this.height })

.appendTo(this.$container)

.get(0)

var rate = this.FOCUS\_POSITION / (this.z + this.FOCUS\_POSITION),

x = this.renderer.width / 2 + this.x \* rate,

y = this.renderer.height / 2 - this.y \* rate;

return { rate: rate, x: x, y: y };

},

re

}

} else {

this.phi += Math.PI / (axis.y == this.thresholdY ? 200 : 500);

this.phi %= Math.PI;

}

if (this.y <= -this.renderer.height \* this.SURFACE\_RATE) {

this.x += 2;

this.y = -this.renderer.height \* this.SURFACE\_RATE;

} else {

this.x += this.vx;

this.y += this.vy;

}

return (

this.z > -this.FOCUS\_POSITION &&

this.z < this.FAR\_LIMIT &&

this.x < this.renderer.width \* 1.5

);

},

};

$(function () {

RENDERER.init();

});

</script>

</body>

</html>