《点线创作系统报告》

1. **设计理念**

在设计这个点线创作系统主要是对我之前设计的随机点线系统的延续，使之能更加自定义地创作出合适地作品。

这里我制作的点线系统的基本属性是：

点是主体，线是对点之间关系地反映，线的透明度反映了点之间的距离（距离过大则无）。

其他的属性包括：

背景（可不变，也可按自定义频率变换）

动态（用户可以使点的运动与鼠标位置相关联，也可以不关联，也可使点不运动）

1. **技术方案**

1核心算法（点线架构）

简而言之就是以CircleArr这个队列存储点circle的信息，因为以一个像素大的面积来表示不够直观，在这里就用小圆circle来表示点。在draw函数中，依据队列里点的信息，用两个循环结构分别画出点和线。

相关代码如下：

for (var j = 0; j < POINT; j++) //画点

{

var cirr=CircleArr[j];

drawCircle(cirr.x,cirr.y,cirr.r,cirr.moveX,cirr.moveY)

}

for (var i = 0; i < POINT; i++) //画线

{

for (var j = 0; j < POINT; j++)

{

if (i + j < POINT)

{

var A = abs(CircleArr[i+j].x - CircleArr[i].x),

B = abs(CircleArr[i+j].y - CircleArr[i].y);

var lineLength = sqrt(A\*A + B\*B);

var C = 1/lineLength\*5-0.02;

var lineOpacity = C > 0.04 ? 0.04 : C;

if (lineOpacity > 0)

{

drawLine(CircleArr[i].x, CircleArr[i].y, CircleArr[i+j].x, CircleArr[i+j].y, lineOpacity);

}

}

}

}

function Line\_O(x1,y1,x2,y2,o)

{

this.beginX = x1,

this.beginY = y1,

this.closeX = x2,

this.closeY = y2,

this.o = o;

}

function Circle (x, y, r, moveX, moveY)

{

this.x = x,

this.y = y,

this.r = r,

this.moveX = moveX,

this.moveY = moveY;

}

function drawCircle(x,y,r,moveX,moveY)

{

var circle = new Circle(x, y, r, moveX, moveY)

noStroke();

color\_cir=color(0, 0, 0,0.3\*255);

fill(color\_cir);

ellipse(x,y,r,r);

return circle;

}

function drawLine(x1,y1,x2,y2,o)

{

var line\_1 = new Line\_O(x1, y1, x2, y2, o)

form = color(0,0,0, 9\*o\*255);

stroke(form);

line(x1, y1, x2, y2);

}

2动态算法

在核心算法机制上加入点的运动性，即每个点在创建使分配一个方向适量，这个矢量包含在点的信息内，在每一次draw（）时，在当前位置加上矢量，形成新的位置，如果位置超出画布，使之在另一边的对应位置出现。

相关代码如下：

if(mouseX-WIDTH/2>0&&Motion\_Is\_Related\_to\_mouse\_position==1)

{

cir.x+=cir.moveX\*W\_SDTM/4;

cir.y+=cir.moveY\*H\_SDTM/4;

}

else

{

cir.x-=cir.moveX\*W\_SDTM/4;

cir.y-=cir.moveY\*H\_SDTM/4;

}

if(cir.x>WIDTH)

cir.x=0;

else if(cir.x<0)

cir.x=WIDTH;

if(cir.y>HEIGHT)

cir.y=0;

else if(cir.y<0)

cir.y=HEIGHT;

}

3背景算法

利用学过的noise函数，使背景颜色不断变化

if(Color\_Is\_Relate\_to\_distance==1)

{

WH=10;

}

else

{

WH=slider.value();

}

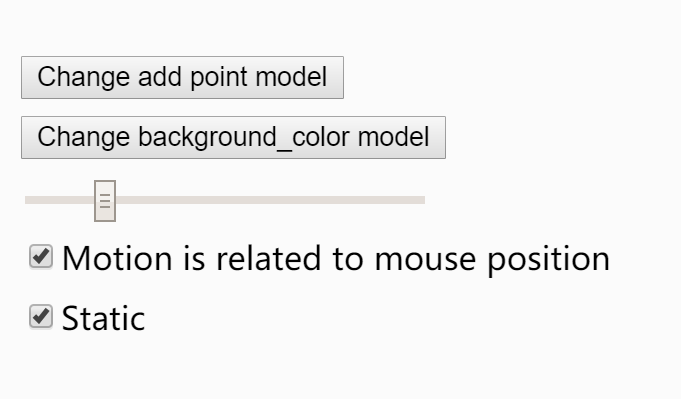
Roff=Roff+0.001\*WH/8;

Goff=Goff+0.002\*WH/8;

Boff=Boff+0.003\*WH/8;

background(255\*noise(Roff),255\*noise(Goff),255\*noise(Boff));

4用户自定义选项



从上到下分别是 1改变加点方式，可以是点击式加单个点，也可以是拖拽式加一堆点.

2改变背景色变化速率，默认速率为10，可以通过下方托条改变速率1-50.

3点运动是否与鼠标位置相关，相关则鼠标离画布中心点的矢量会改变点的运动方向和速度.

4点可以是精致的，这便于静态创作。

相关代码如下

1加点：

button\_1 = createButton('Change add point model');

button\_1.position(WIDTH+19, 30);

button\_1.mousePressed(changeAP);

function changeAP() { //change add point model

IsmousePressed=-IsmousePressed;

}

function mouse\_IN\_canvous()

{

var IS\_in;

if(mouseX<WIDTH&&mouseX>0&&mouseY<HEIGHT&&mouseY>0)

{

IS\_in=1;

}

else

{

IS\_in=0;

}

return IS\_in;

}

function mousePressed() {

if(IsmousePressed==1&&mouse\_IN\_canvous())

{

for (var j = 0; j < AddPoint; j++)

{

CircleArr.push(drawCircle(mouseX,mouseY,random(6,2),random(1/2,-1/2),random(1/2,-1/2)));

POINT++;

}

}

}

function mouseDragged() {

if(IsmousePressed==-1&&mouse\_IN\_canvous())

{

for (var j = 0; j < AddPoint; j++)

{

CircleArr.push(drawCircle(mouseX,mouseY,random(6,2),random(1/2,-1/2),random(1/2,-1/2)));

POINT++;

}

}

}

2背景色：

button\_2 = createButton('Change background\_color model');

button\_2.position(WIDTH+19, 30\*2);

button\_2.mousePressed(changeBC);

slider = createSlider(1, 50, 10);

slider.position(WIDTH+19, 30\*3);

slider.style('width', '200px');

function changeBC() { //change add point model

Color\_Is\_Relate\_to\_distance=-Color\_Is\_Relate\_to\_distance;

}

if(Color\_Is\_Relate\_to\_distance==1)

{

WH=10;

}

else

{

WH=slider.value();

}

Roff=Roff+0.001\*WH/8;

Goff=Goff+0.002\*WH/8;

Boff=Boff+0.003\*WH/8;

background(255\*noise(Roff),255\*noise(Goff),255\*noise(Boff));

3鼠标相关/静态

checkbox\_1 = createCheckbox('Motion is related to mouse position', true);

checkbox\_1.position(WIDTH+19, 30\*4);

checkbox\_1.changed(myChecked\_1Event);

checkbox\_2 = createCheckbox('Static', false);

checkbox\_2.position(WIDTH+19, 30\*5);

checkbox\_2.changed(myChecked\_2Event);

function myChecked\_1Event()

{

Motion\_Is\_Related\_to\_mouse\_position=-Motion\_Is\_Related\_to\_mouse\_position;

}

function myChecked\_2Event()

{

Is\_Static=-Is\_Static;

}

var W\_SDTM,H\_SDTM; //Width\_SqrtDistanceToMid

if(Motion\_Is\_Related\_to\_mouse\_position==1)

{

W\_SDTM=sqrt(abs(randomGaussian(mouseY-WIDTH/2,0)));//1-17.4

H\_SDTM=sqrt(abs(randomGaussian(mouseY-HEIGHT/2,0)));//1-14.1

} //WH=sqrt(W\_SDTM\*W\_SDTM+H\_SDTM\*H\_SDTM);//1.4-22.5

else

{

W\_SDTM=2;

H\_SDTM=2;

}

if(Is\_Static==1)

{

W\_SDTM=0;

H\_SDTM=0;

}

if(mouseX-WIDTH/2>0&&Motion\_Is\_Related\_to\_mouse\_position==1)

{

cir.x+=cir.moveX\*W\_SDTM/4;

cir.y+=cir.moveY\*H\_SDTM/4;

}

else

{

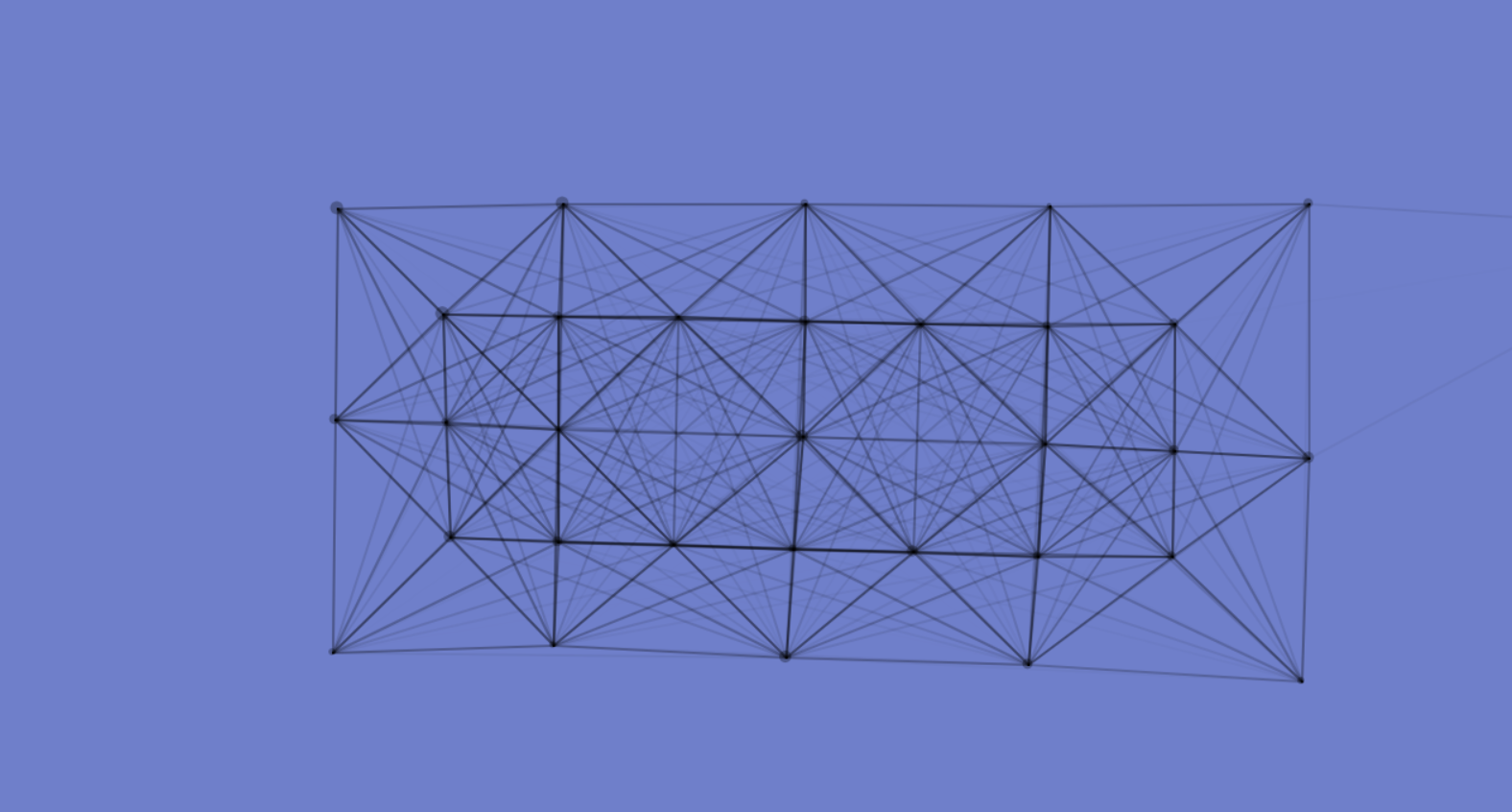
cir.x-=cir.moveX\*W\_SDTM/4;

cir.y-=cir.moveY\*H\_SDTM/4;

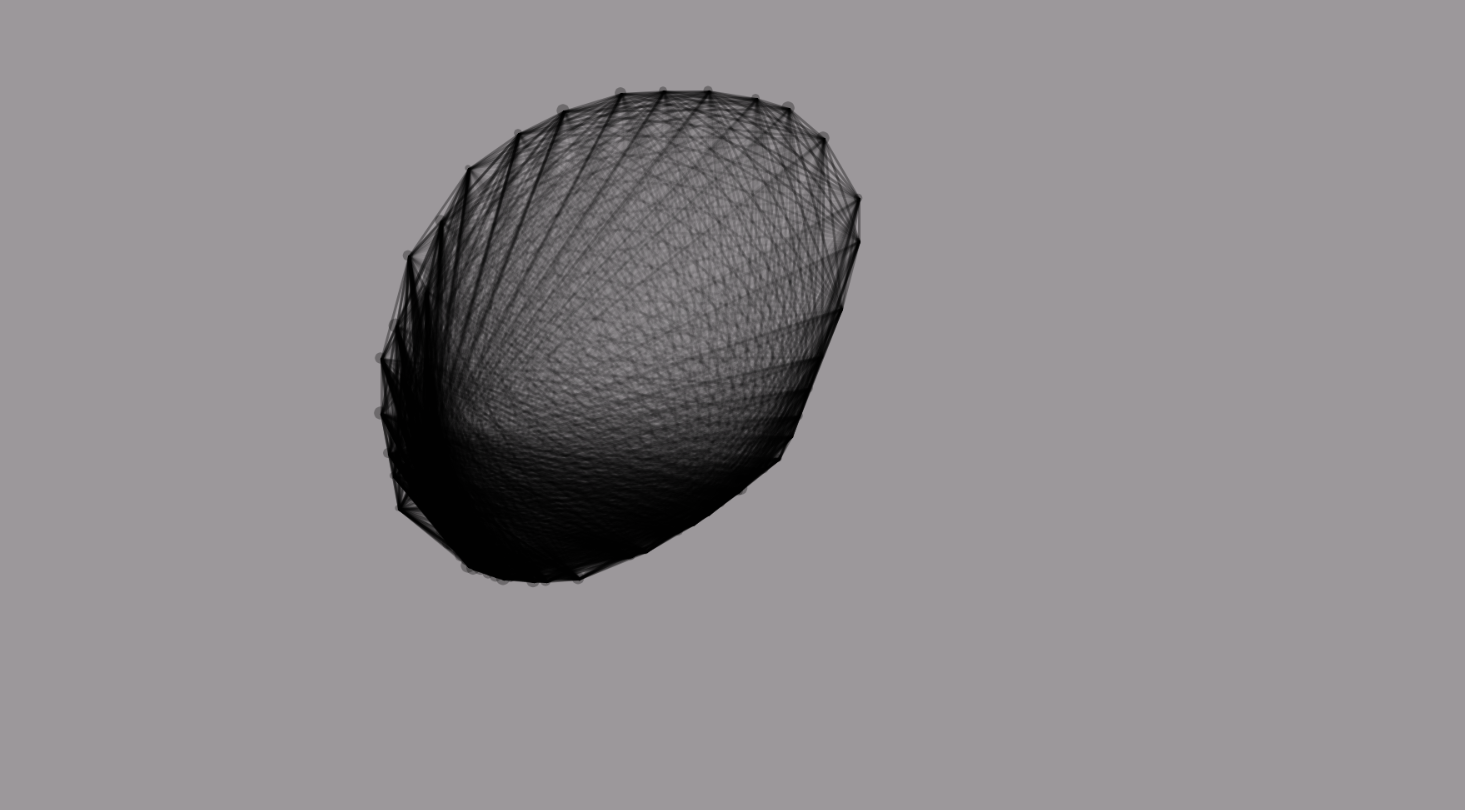
}

1. **示例作品（静态）**

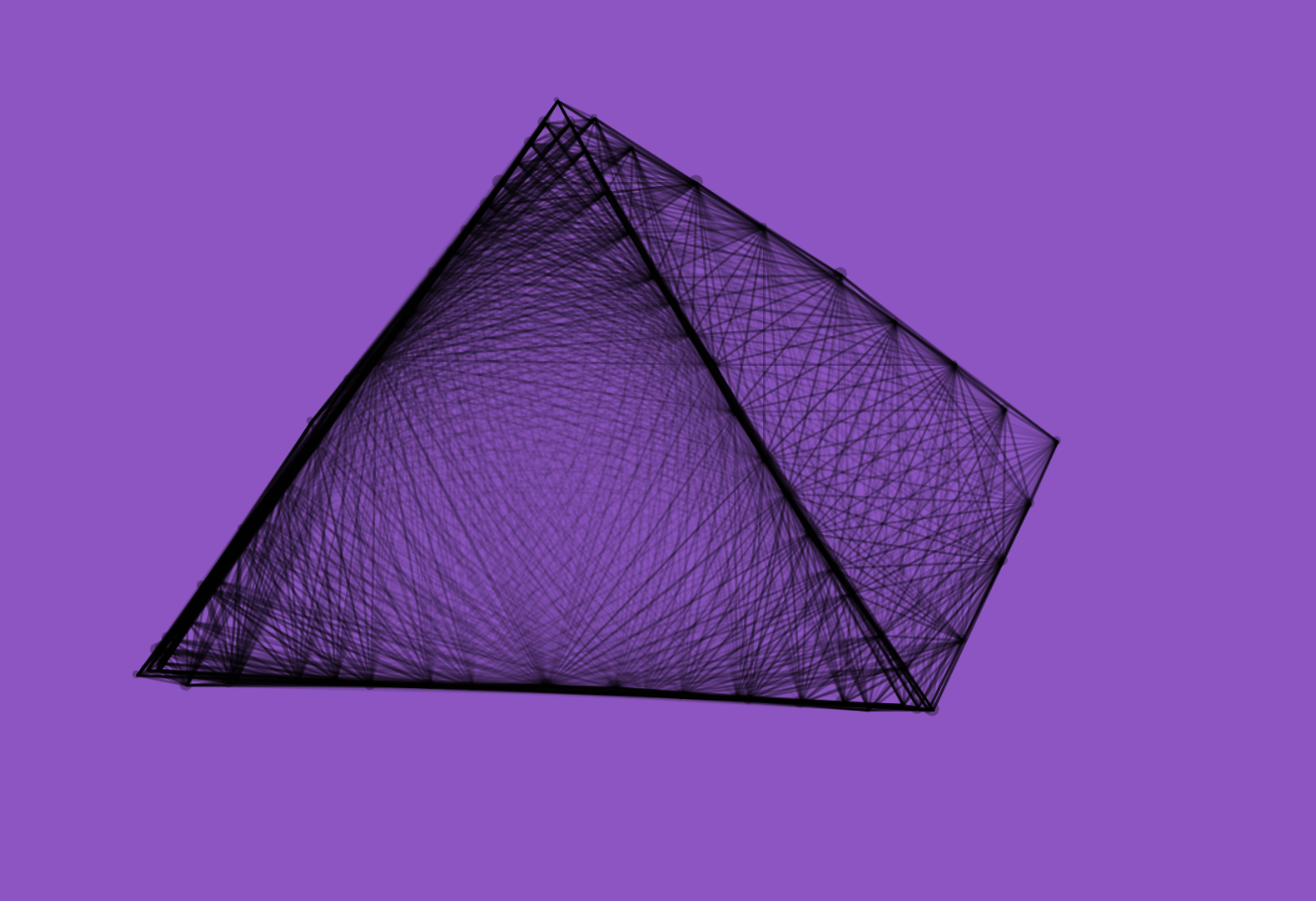
**1、3\*5点阵**



**2贝壳**



**3正五面体**



1. **参照对象介绍**

**1一开始看的是知乎登陆界面，然后又参考了一些类似的效果，如下图所示。借鉴了其两点之间距离越短，透明度越低的思路。**

