flowfieldforap

```
float noisescale = 0.001;
int spawnheight = 200;
//height of spawn location
int spawnwidth = 200;
//width of spawn location
int howmany = 500;
//how many particles
int b = 0;
int t = 0;

class Particle{
 //have
 float x;
 //x position
 float y;
 //y position
 float speed;
 //speed
 float direction;
 //direction
 color col;
 //color
 float radius;
 //radius
 float acceleration;
 //acceleration
 float mousex = 0;
 //xeffect
 float mousey = 0;
 //yeffect

 Particle(color c, float s){
 col = c;
 speed = s;
 //set color and speed to input from setup
 x = 0.5*displayWidth + floor(random(-spawnwidth*0.5, spawnwidth*0.5));
 //move x to center of screen, move x right/left by up to half of spawn width
 y = 0.5*displayHeight + floor(random(-spawnheight*0.5, spawnheight*0.5));
 //same thing with y
 direction = random(TWO_PI);
 //give each particle random direction based on an angle on the unit circle
 radius = 10;
 }

 void display(){
 fill(col);
 circle(x, y, radius);
```

```
 //fill circle and display circle at x and y position
 }

 void update(){
  //check if particle is still on screen
  if ((x > 0) && (x < displayWidth) && (y < displayHeight) && (y > 0)) {
  if (mouseX > displayWidth/2){
   mousex = 0.2;
  }else{
   mousex = -0.2;
  }
  //change xeffect based on mouse position
  if (mouseY > displayHeight/2){
   mousey = 0.2;
  }else{
   mousey = -0.2;
  }
  //same for y
  acceleration = map(noise(x*noisescale, y*noisescale, frameCount*noisescale), 0.4, 0.6, 0, 1);
  acceleration = acceleration * TWO_PI;
  //update acceleration using noise based on x/y position and frame count
  direction = 0.98*direction + 0.02*acceleration;
  //update direction using 98% previous direction and 2% new direction
  x = x + (cos(direction) * speed + mousex);
  //update x position
  y = y + (sin(direction) * speed+ mousey);
  //update y position
  } else{
    reset();
    //if not on screen, reset
  }
 }
 void reset(){
 //if not on screen,
  x = 0.5*displayWidth + floor(random(-spawnwidth*0.5, spawnwidth*0.5));
  y = 0.5*displayHeight + floor(random(-spawnheight*0.5, spawnheight*0.5));
  //reset position to spawn area
  direction = random(TWO_PI);
  //reset direction
 }
}

Particle[] particles;
//set particles = to a list of Particles

void setup(){
 noStroke();
 size(displayWidth, displayHeight);
```

```
 background(0);

 particles = new Particle[howmany];
 //add particles to the list of particles
 for (int i = 0; i < floor(howmany/3); i++){
   particles[i] = new Particle(color(255, 57, 31), 0.4);
 }
 //set one third of the particles to a red color and 0.4 speed
 for (int i = floor(howmany/3); i < floor(howmany/3)+(howmany/3); i++){
   particles[i] = new Particle(color(57, 255, 20), 0.6);
 }
 //one third to a green color and 0.6 speed
 for (int i = floor(howmany/3)+(howmany/3); i < howmany; i++){
   particles[i] = new Particle(color(255, 255, 255), 0.8);
 }
 //set one third+1 of the particles to a white color and 0.8 speed
}

void draw(){
 changebackground(40);
 //at each frame, display and update each particle
  for (int i = 0; i < howmany; i++){
  particles[i].display();
  particles[i].update();
   }
}

void mouseClicked(){
//when mouse clicked, clear screen and reset particles
clear();
for (int i = 0; i < howmany; i++){
  particles[i].reset();
}
}

void changebackground(int sp){
 background(b);
 if ((t < 1) && (b < 175)){
 b++;
 } else {
 t = 1;
 b--;
 }
 if (b < 1){
  t = 0;
 }
 for (int x = 0; x < displayWidth; x = x + sp){
   fill(random(255), random(255), random(255));
```

```
    for (int y = 0; y < displayHeight; y = y + sp){
      circle(x, y, 5);
  }
  }
}
```