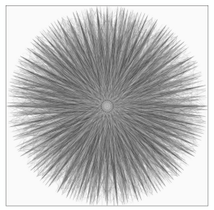
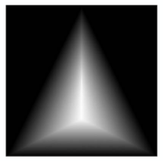
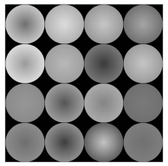


```
Poly Pattern III (polystar)
/* Poly Pattern III ira Greenberg, November 26, 2005 revised October 18, 2006 Pattern:
Polystar */
void setup(){
size(400, 400);
background(randomGS5());
smooth();
float ang = 0;
int steps = int(random(30, 200));
for (int i=0; i<steps; i++){
makePoly(int(width/2), height/2, int(random(3, 10)), Â
ang+=360/steps, random(-width/2, width/2), Â random(-width/2, width/2), int(random(255),
Â random(25, 0), fill[0]);
}}
void makePoly(float x, float y, int points, float initAngle, Â
float radius, float radius2, int strokeCol, float strokeWt, Â int stroke){
float px=0, py=0; float angle = initAngle; float initRadius = radius; float halfRadius = radius2;
stroke(stroke); noFill(); strokeJoin(strokeJoin); strokeWeight(strokeWt); beginShape(); // //
2nd ratios create polystar // else create regular poly if radius2 !=0){
points*=2; } for (int i=0; i<=points; i++){
//alternates radius length if polystar
if (radius2 != 0 && i%2 == 0){
radius = halfRadius; } else {
radius = initRadius; } px = x+cos(radians(angle))*radius;
py = y+sin(radians(angle))*radius; vertex(px, py); angle+=360/points;
}
endShape(CLOSE);
}
```



```
//Triangle Blur
Point3[]a = new Point3[3]; float shift = 2; float fade = 0;
float fillCol = 0; void setup(){
size(400, 400);
background(0);
smooth();
fade = 255.0/(width/2.0/shift);
p0[0] = new Point3; height-1);
p1[1] = new Point3(width-1, height-1);
p2[2] = new Point3(width/2, 1);
noStroke();
triBlur();
}
void triBlur(){
fill(fillCol);
fillCol+=fade;
triangle(p0[0].x+shift, p0[].y+shiftV2, p[1].x+shift, Â
p[1].y+shiftV2, p[2].x, p[2].y+shift);
if(p[2].x==width/2){
// recursive call
triBlur(); }
}
```



```
// Simple Radial Gradient // January 12, 2006 void setup(){
size(400, 400);
background(0);
smooth();
// create a simple table of gradients int columns = 4; int radius = (width/columns)/2; // create
some gradients for (int i=radius; i<width; i+=radius*2){
for (int j=radius; j<height; j+=radius*2){
createGradient(i, j, radius, Â
color(int(random(255)), int(random(255)), int(random(255)), Â
color(int(random(255)), int(random(255)), int(random(255))a
);}
}
void createGradient (float x, float y, float radius, Â
color c1, color c2){
float px = 0, py = 0, angle = 0;
// calculate differences between color components float deltaR = red(c2)-red(c1); float
deltaG = green(c2)-green(c1); float deltaB = blue(c2)-blue(c1); // hack to ensure there are no
holes in gradient
```

```
// needs to be increased, as radius increases float gapFiller = 8.0;
for (int i=0; i<radius; i+=4){
for (float j=0; j<360; j+=1.0/gapFiller){
px = x+cos(radians(angle))*i; py = y+sin(radians(angle))*i; angle+=1.0/gapFiller; color c =
color( Â
(red(c1)+(i*(deltaR/radius))), Â (green(c1)+(i*(deltaG/radius))), Â (blue(c1)+(i*(deltaB/radius))),
Â
set(int(px), int(py), c);
} // adds smooth edge // hack anti-aliasing
noFill();
strokeWeight(3);
ellipse(x, y, radius*2, radius*2);
}
```



```
/*
Bézier Ellipse
Ira Greenberg, December 19, 2005
Revised: November 15, 2006
*/
// arrays to hold ellipse coordinate data float[] px, py, cx, cy, cx2, cy2;
// global variable - points in ellipse int pts = 4;
color controlPtCol= #222222; color anchorPtCol = #BBBBBB;
void setup(){
size(500, 600);
setEllipse(pts, 130, 130); }
void draw(){
background(145);
drawEllipse(); }
// draw ellipse with anchor/control points void drawEllipse(){
strokeWeight(1.125);
stroke(255);
noFill();
// create ellipse
for (int i=0; i<pts; i++){
if (i==pts-1) {
bezier(px[i], py[i], cx[i], cy[i], cx2[i], cy2[i], Â
px[0], py[0]);
} else{
bezier(px[i], py[i], cx[i], cy[i], cx2[i], cy2[i], Â
px[i+1], py[i+1]);
} }
```

```
strokeWeight(.75); stroke(0); rectMode(CENTER);
// control handles and tangent lines for (int i=0; i<pts; i++){
if (i==pts-1){// last loop iteration - close path
line(px[0], py[0], cx2[i], cy2[i]); } if (i>0){
line(px[i], py[i], cx2[i-1], cy2[i-1]); } line(px[i], py[i], cx[i], cy[i]);
}
for (int i=0; i<pts; i++){
fill(controlPtCol);
noStroke();
//control handles
ellipse(cx[i], cy[i], 4, 4);
ellipse(cx2[i], cy2[i], 4, 4);
fill(anchorPtCol); stroke(0); //anchor points rect(px[i], py[i], 5, 5);
}
}
// fill up arrays with ellipse coordinate data void setEllipse(int points, float radius, float
controlRadius){
pts = points;
px = new float[points];
py = new float[points];
cx = new float[points];
cy = new float[points];
cx2 = new float[points];
cy2 = new float[points];
float angle = 360.0/points;
float controlAngle1 = angle/3.0;
float controlAngle2 = controlAngle1*2.0;
for (int i=0; i<points; i++){
px[i] = width/2+cos(radians(angle))*radius;
py[i] = height/2+sin(radians(angle))*radius; cx[i] =
width/2+cos(radians(angle+controlAngle1))* Â
controlRadius+cos(radians(controlAngle1)); cy[i] =
height/2+sin(radians(angle+controlAngle1))* Â
controlRadius+sin(radians(controlAngle1)); cx2[i] =
width/2+cos(radians(angle+controlAngle2))* Â
```

```
controlRadius+cos(radians(controlAngle1)); cy2[i] =
height/2+sin(radians(angle+controlAngle2))* Â
controlRadius+sin(radians(controlAngle1));
//increment angle so trig functions keep chugging along angle+=360.0/points;
}
}
void mousePressed(){
setEllipse(int(random(0, 36)), random(-200, 300), random(-200, 300)); }
```



```
//Triangle Flower Point[]p = new Point[3]; float shift = 1.0; float fade = 0; float fillCol = 0;
float rot = 0;
float spin = 0;
void setup(){
size(400, 400);
background(0);
smooth();
fade = 255.0/(width/2.0/shift);
spin = 360.0/(width/2.0/shift);
p0[0] = new Point(-width/2, height/2);
p[1] = new Point(width/2, height/2);
p[2] = new Point(0, -height/2);
noStroke();
translate(width/2, height/2);
triBlur();
}
void triBlur(){
fill(fillCol);
fillCol+=fade;
rotate(spin);
// another interesting variation: uncomment the line bellow
// rotate(rot+=radians(spin));
triangle(p0[0].x+shift, p0[].y+shiftV2, p[1].x+shift, Â
p[1].y+shiftV2, p[2].x, p[2].y+shift);
if(p[2].x==0){
// recursive call
triBlur(); }
}
```



```
*/Wave Gradient
January 13, 2006
*/

size(600, 400);
background(200,200,200);
float angle = 0; float px = 0, py = 0;
float amplitude = 30;
float frequency = 0;
float fillGap = 2.5;
color c;

for (int i=-50; i<height+50; i+=){
// reset angle to 0, so waves stack properly
angle = 0;
// increasing frequency causes more gaps
frequency+=.003;
for (float j=0; j<width+50; j+=){
py = i+sin(radians(angle))*amplitude; angle+=frequency; c = color( Â
abs(py-i)*255/amplitude, Â 255-abs(py-i)*255/amplitude, Â 255-abs(py-i)*255/width+50) );
// hack to fill gaps. Raise value of fillGap // if you increase frequency for (int fillx=0;
fillx<fillGap; fillx++){
set(int(j-fillx), int(py)-fillar, c);
set(int(j), int(py), c);
set(int(j+fillx), int(py)+fillar, c); }
} }
/*
```



```
Left click to switch colors
Hold middle-click to make it disappear
Right-click to toggle shader
```

```
// Notes:
  This may not run on certain mobiles or devices with weaker gpus.
  If you get an error about max uniforms try decreasing `PARTICLE_MAX_COUNT` and `TRAIL_MAX_COUNT`
*/

// Values to play with.
const TRAIL_MAX_COUNT = 30;
const TRAIL_MAX_RADIUS = 1.20;
const PARTICLE_MAX_COUNT = 70;
const PARTICLE_PULL_FORCE = 0.0015;
const PARTICLE_INITIAL_SPEED = 0.15;
const PARTICLE_SPEED_VARIANCE = 15;
const PARTICLE_SPAWN_ANGLE = 25;
const PARTICLE_LIFE = 60;
const PARTICLE_MIN_AIR_DRAG = 0.9;
const PARTICLE_MAX_AIR_DRAG = 0.95;

var myFont;
var theShader;
var shaderTexture;
var trail;
var shaded = true;
var timeSteps = 1;
var sketchTimer = 0;
var transitionTimer = 0;
var globalHue = 0;

function preload() {
  theShader = new p5.Shader(this.renderer, vertShader, fragShader);
  myFont = loadFont('Roboto-Black.ttf');
}

function setup() {
  pixelDensity(1);

  let canvas = createCanvas(
    min(windowWidth, windowHeight),
    min(windowWidth, windowHeight),
    WEBGL);

  canvas.canvas.oncontextmenu = () => false;  // Removes right-click menu.

  globalHue = random(255);

  trail = new Trail();

  shaderTexture = createGraphics(width, height, WEBGL);
  shaderTexture.noStroke();
}

function draw() {
  background(0);
  noStroke();

  translate(-width / 2, -height / 2);

  let lerpAmount = 0.1;

  if (transitionTimer == 0) {
    // Normal motion.
    timeSteps = lerp(timeSteps, 1, lerpAmount);
    trail.radius = lerp(trail.radius, TRAIL_MAX_RADIUS, lerpAmount);
  } else {
    // Spin faster in a tigher circle.
    transitionTimer--;
    timeSteps = lerp(timeSteps, 4, lerpAmount);
    trail.radius = lerp(trail.radius, TRAIL_MAX_RADIUS * 0.5, lerpAmount);
  }

  trail.hue = lerp(trail.hue, globalHue, lerpAmount * 0.5);

  trail.move();

  if (shaded) {
    trail.display();
  } else {
    trail.debug(Display);
  }

  sketchTimer += timeSteps;

  fill(255);
  noStroke()
  textFont(myFont);
  text(`
    Left-click to switch colors
    Hold middle-click to make it disappear
    Right-click to toggle shader`,
    40, 40);
}

function mousePressed() {
  if (mouseButton == LEFT) {
    globalHue = random(255);
    transitionTimer = 40;
  } else if (mouseButton == RIGHT) {
    shaded = !shaded;
  }
}
```



```
void setup(){
size(400, 400);
// fill display window with default black and white // linear gradient background
LinearGradient lg = new LinearGradient(0, lg.create);
//create 4 linear Gradients color c1 = color(255, 0, 0); color c2 = color(0, 255, 0);
color c3 = color(255, 255, 0); color c4 = color(180, 0, 255);
color c5 = color(255, 127, 0); color c6 = color(0, 0, 255);
color c7 = color(0, 255, 255); color c8 = color(255, 180, 0);
Rectangle r1 = new Rectangle(50, 50, 150, 150); Rectangle r2 = new Rectangle(200, 50, 150,
150); Rectangle r3 = new Rectangle(50, 200, 150, 150); Rectangle r4 = new Rectangle(200,
200, 150, 150);
LinearGradient lg1 = new LinearGradient(r1, c2, r1, Â
GradientLAXIS_HORIZONTAL);
lg1.create(); LinearGradient lg2 = new LinearGradient(r3, c4, r2, lg2.create(); LinearGradient
lg3 = new LinearGradient(r5, c6, r3); lg3.create(); LinearGradient lg4 = new
LinearGradient(r7, c8, r4, Â
Gradient.AXIS_HORIZONTAL);
lg4.create();
//create 4 radial Gradients r1 = color(0, 150, 200); c2 = color(200, 200, 100);
c3 = color(190, 225, 290); c4 = color(90, 45, 20);
c5 = color(195, 195, 90); c6 = color(30, 10, 70);


c7 = color(0); c8 = color(255);
r1 = new Rectangle(60, 60, 130, 130); r2 = new Rectangle(225, 60, 150, 130); r3 = new
Rectangle(60, 225, 130, 150); r4 = new Rectangle(210, 210, 130, 130);
RadialGradient rg1 = new RadialGradient(r1, c2, r1); rg1.create();
RadialGradient rg2 = new RadialGradient(r3, c4, r2); rg2.create();
RadialGradient rg3 = new RadialGradient(r5, c6, r3); rg3.create();
RadialGradient rg4 = new RadialGradient(r7, c8, r4); rg4.create();
}
```



```
size(500,500);
background(181,176,157);

//0 Minutes
stroke(0, 48, 143);
strokeWeight(25);

//left-bottom
point(map(10,0,100,0,500),map(90,0,100,0,500));
point(map(32,0,100,0,500),map(80,0,100,0,500));

point(map(15,0,100,0,500),map(80,0,100,0,500));
point(map(15,0,100,0,500),map(75,0,100,0,500));
point(map(15,0,100,0,500),map(70,0,100,0,500));

point(map(20,0,100,0,500),map(80,0,100,0,500));
point(map(20,0,100,0,500),map(75,0,100,0,500));
point(map(20,0,100,0,500),map(70,0,100,0,500));

//right-bottom
point(map(80,0,100,0,500),map(90,0,100,0,500));
point(map(80,0,100,0,500),map(80,0,100,0,500));
//bodyball left
point(map(65,0,100,0,500),map(70,0,100,0,500));
point(map(65,0,100,0,500),map(75,0,100,0,500));
point(map(63,0,100,0,500),map(80,0,100,0,500));
//bodyball right
point(map(75,0,100,0,500),map(70,0,100,0,500));
point(map(80,0,100,0,500),map(75,0,100,0,500));
point(map(75,0,100,0,500),map(80,0,100,0,500));

//left top
point(map(10,0,100,0,500),map(10,0,100,0,500));
point(map(7,0,100,0,500),map(15,0,100,0,500));
point(map(20,0,100,0,500),map(13,0,100,0,500));

point(map(20,0,100,0,500),map(13,0,100,0,500));

point(map(32,0,100,0,500),map(13,0,100,0,500));
point(map(40,0,100,0,500),map(13,0,100,0,500));
point(map(46,0,100,0,500),map(9,0,100,0,500));

//right top
point(map(80,0,100,0,500),map(12,0,100,0,500));
point(map(87,0,100,0,500),map(16,0,100,0,500));
//bodyball left
point(map(55,0,100,0,500),map(37,0,100,0,500));
point(map(55,0,100,0,500),map(45,0,100,0,500));
//bodyball right
point(map(77,0,100,0,500),map(36,0,100,0,500));

//10 Minutes
stroke(16, 106, 210);
strokeWeight(15);

//left-bottom
point(map(10,0,100,0,500),map(90,0,100,0,500));
point(map(27,0,100,0,500),map(90,0,100,0,500));
point(map(32,0,100,0,500),map(90,0,100,0,500));
point(map(37,0,100,0,500),map(90,0,100,0,500));

point(map(15,0,100,0,500),map(80,0,100,0,500));
point(map(15,0,100,0,500),map(75,0,100,0,500));
point(map(15,0,100,0,500),map(70,0,100,0,500));

point(map(20,0,100,0,500),map(80,0,100,0,500));
point(map(20,0,100,0,500),map(75,0,100,0,500));
point(map(20,0,100,0,500),map(70,0,100,0,500));

//right-bottom
point(map(87,0,100,0,500),map(87,0,100,0,500));
//bodyball left
point(map(53,0,100,0,500),map(73,0,100,0,500));
point(map(67,0,100,0,500),map(75,0,100,0,500));
point(map(53,0,100,0,500),map(83,0,100,0,500));
//bodyball right
point(map(85,0,100,0,500),map(70,0,100,0,500));
point(map(75,0,100,0,500),map(75,0,100,0,500));
point(map(85,0,100,0,500),map(80,0,100,0,500));

//left top
point(map(14,0,100,0,500),map(10,0,100,0,500));
point(map(18,0,100,0,500),map(10,0,100,0,500));

point(map(13,0,100,0,500),map(15,0,100,0,500));
point(map(16,0,100,0,500),map(15,0,100,0,500));
point(map(10,0,100,0,500),map(15,0,100,0,500));

point(map(20,0,100,0,500),map(13,0,100,0,500));

point(map(32,0,100,0,500),map(13,0,100,0,500));
point(map(40,0,100,0,500),map(13,0,100,0,500));
point(map(46,0,100,0,500),map(9,0,100,0,500));

//right top
point(map(80,0,100,0,500),map(12,0,100,0,500));
point(map(87,0,100,0,500),map(16,0,100,0,500));
//bodyball left
point(map(65,0,100,0,500),map(40,0,100,0,500));
point(map(65,0,100,0,500),map(40,0,100,0,500));
//bodyb
```