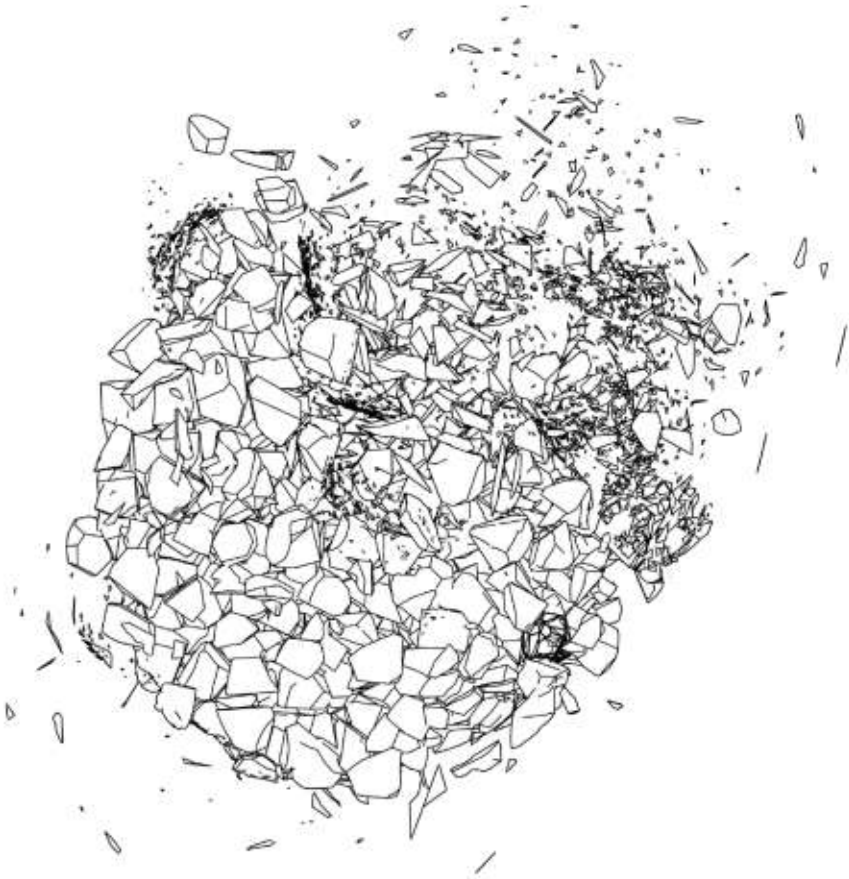
COOKIE COLLECTIVE

NUMBER 1 FEBRUARY 2019



COLLECTION OF SHADERS

Shaders are like poetry.
It is a codified text that only makes sense when one knows the rules that govern it. As an alexandrine or a haiku, a shader is a composition of words arranged by technical and artistic constraints.

Shaders are like cooking.
The text is a recipe with instructions that blend variables and other ingredients. The shader is then baked in oven to satisfy the appetite of the graphic card.

This book presents visuals generated by shaders accompanied by an extract of their code. The two next double pages contain functions that are common bases for the following shaders, where the code focuses on the crunchy part.

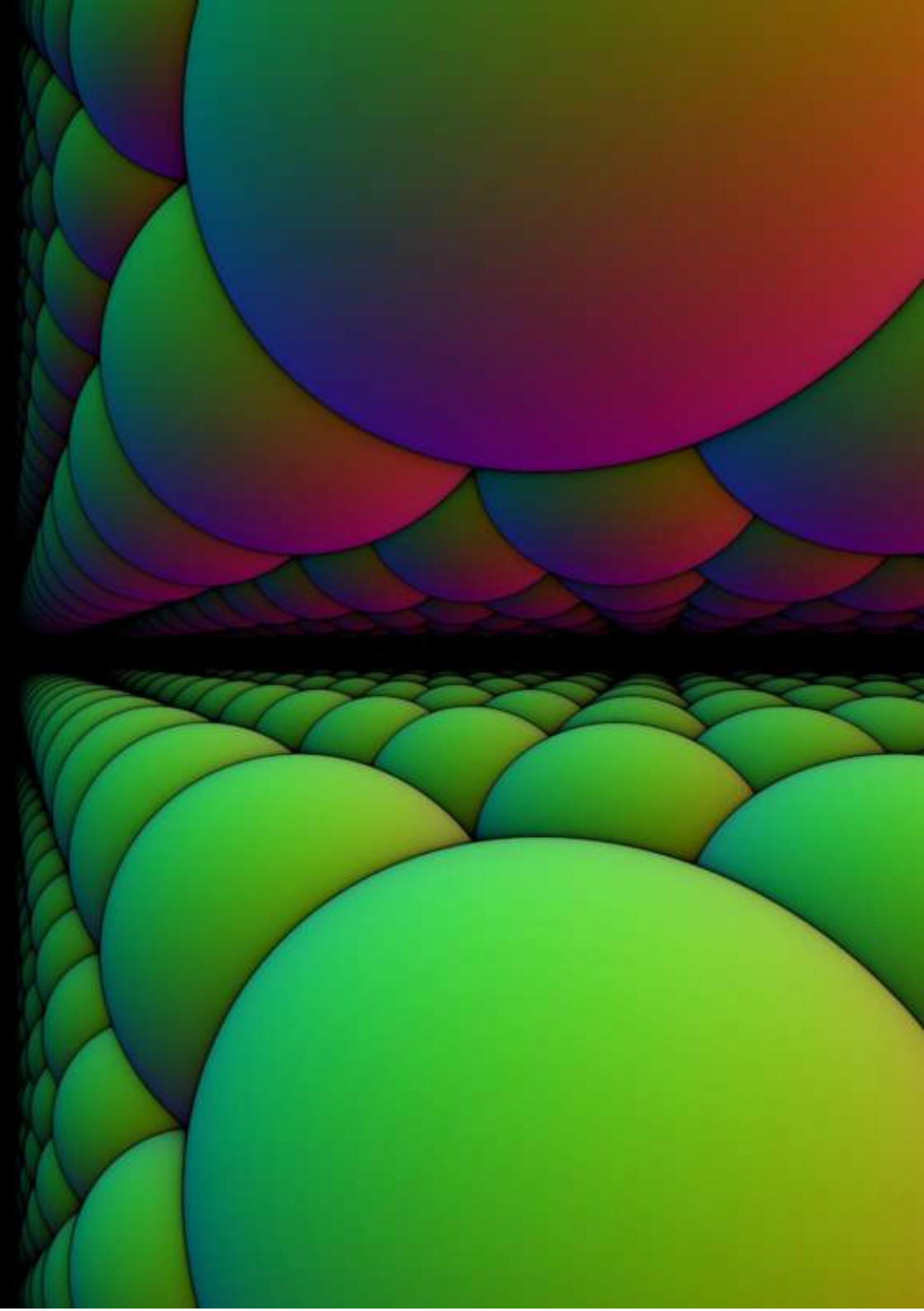
The full text can be accessed through the link on **ShaderToy.com** at the bottom of the page. Watch out for uppercase and lowercase, it's important!

The Cookie Collective exists thanks to the demoscene community. This fanzine presents works from members of the collective.

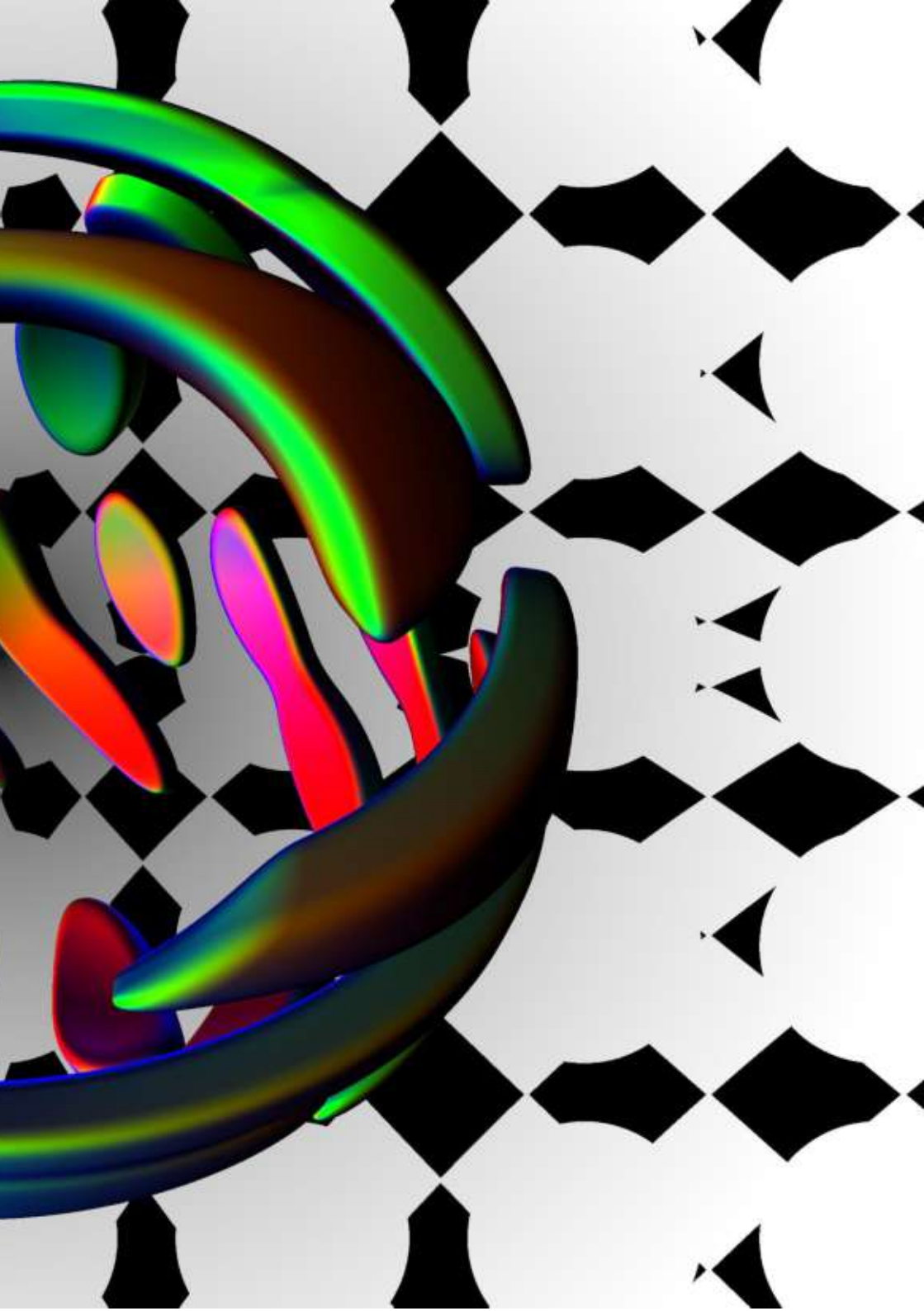
```
float hash(float n) { return fract(sin(n) * 1e4); }
float noise(vec3 x) {
  const vec3 step = vec3(110, 241, 171);
  vec3 i = floor(x);
  vec3 f = fract(x);
  float n = dot(i, step);
  vec3 u = f * f * (3.0 - 2.0 * f);
  return mix(mix(mix(hash(n + dot(step, vec3(0,0,0))),
                     hash(n + dot(step, vec3(1,0,0))),
                     u.x),
                 mix(hash(n + dot(step, vec3(0,1,0))),
                     hash(n + dot(step, vec3(1,1,0))),
                     u.x), u.y),
             mix(mix(hash(n + dot(step, vec3(0,0,1))),
                     hash(n + dot(step, vec3(1,0,1))),
                     u.x),
                 mix(hash(n + dot(step, vec3(0,1,1))),
                     hash(n + dot(step, vec3(1,1,1))),
                     u.x), u.y), u.z);
}

float fbm(vec3 p) {
  float amplitude = .5;
  float result = 0.;
  for (float index = 0.; index <= 4.0; ++index) {
    result += noise(p / amplitude) * amplitude;
    amplitude /= 2.;
  }
  return result;
}

void mainImage(out vec4 fragColor, in vec2 fragCoord) {
  vec2 uv = (fragCoord - .5*iResolution.xy)
    / iResolution.y;
  vec3 seed = vec3(uv * 4.0, 0);
  float salty = fbm(seed * 40.0);
  float curry = fbm(seed + vec3(noise(seed * 2.) * .5));
  fragColor = vec4(curry - salty * .1);
}


// Noise Example using the work of Morgan McGuire
// https://www.shadertoy.com/view/tdXXWn
```

```
float map(vec3 pos) {
  float cell = 3.;
  pos = mod(pos, cell) - cell * .5;
  return length(pos) - 1.;
}

vec3 getNormal (vec3 pos) {
  vec2 e = vec2(1.0, -1.0) * 0.5773 * 0.0005;
  return normalize(e.xyy * map(pos + e.xyy) +
                   e.yyx * map(pos + e.yyx) +
                   e.yxy * map(pos + e.yxy) +
                   e.xxx * map(pos + e.xxx));
}

void mainImage(out vec4 fragColor, in vec2 fragCoord) {
  vec2 uv = (fragCoord - .5*iResolution.xy)
    / iResolution.y;
  vec3 eye = vec3(0, 0, -2);
  vec3 ray = normalize(vec3(uv, 1.));
  float shade = 0.;
  for (float index = 100.; index > 0.; --index) {
    float dist = map(eye);
    if (dist < 0.001) {
      shade = index / 100.;
      break;
    }
    eye += ray * dist;
  }
  vec3 normal = getNormal(eye) * .5 + .5;
  fragColor = vec4(normal * shade, 1);
}
```

```
// Raymarching Example using the work of Inigo Quilez
// https://www.shadertoy.com/view/tsXXWn
```

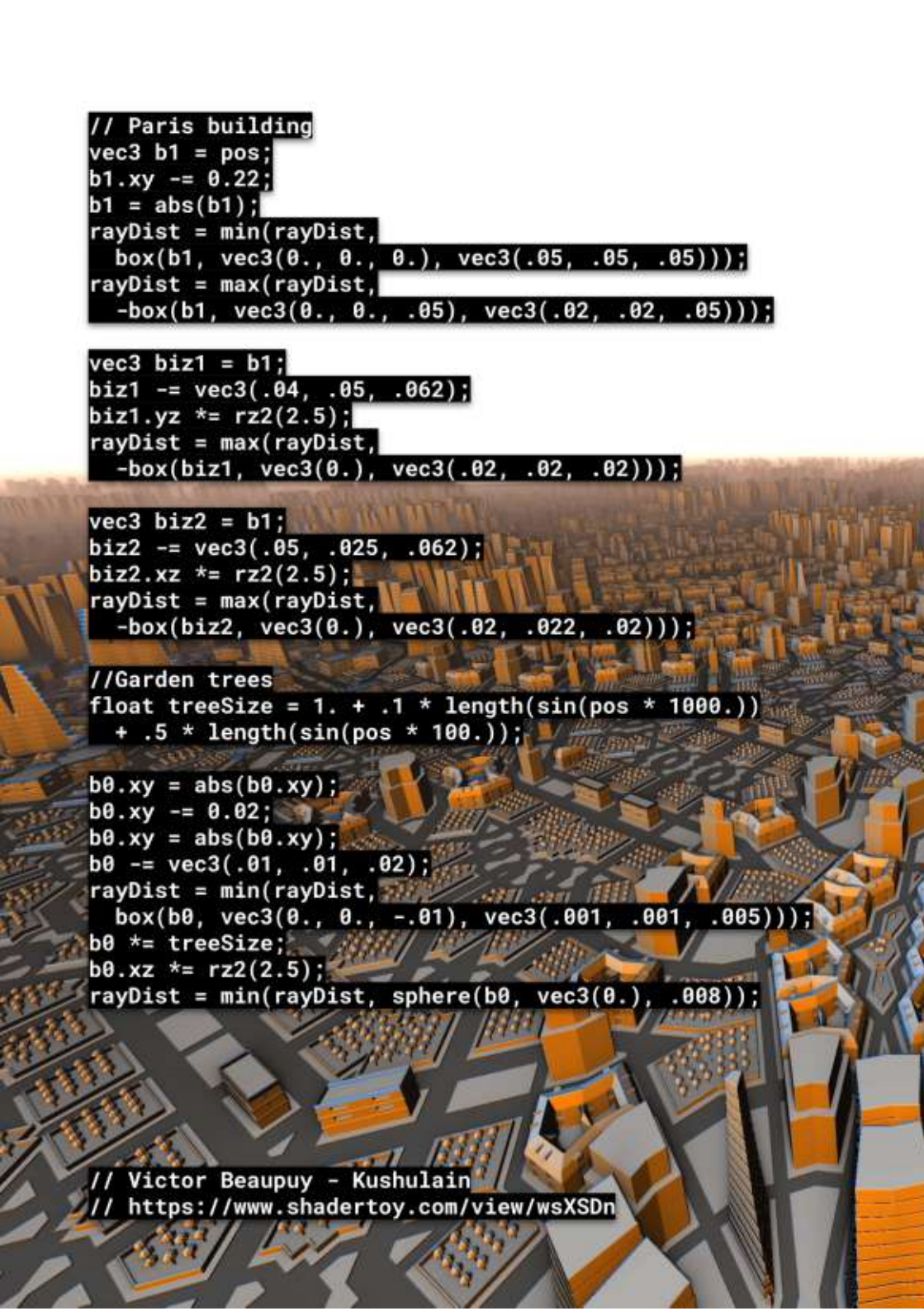


```
vec3 cp = p;
float dist = 1000.;
float time = TIME * .25;
p *= .36;
p.zx *= rot(-time * .25);
p.xz *= rot(p.z*1.1);

for (float it = 0.; it < 2.; it += 1.) {
  p.xz *= rot(sin(p.y + time + fract(sin(it * 2369.)))
    * PI / (it + 1.) * .5);
  p.zy *= rot(time+PI);
  vec3 ap = max(vec3(0.), abs(p)) - 1.;
  float cu = length(ap) - .5
    + min(max(ap.x, max(ap.y, ap.z)), 0.);
  dist = smin(dist, cu , .25);
}

p = cp;
p.xz *= rot(-time * .5);

float rad = 5.;
float wi = .5;
float sph = length(p) - rad + wi /2.;

dist = -smin(sph, -dist, .15);
sph = length(p) - rad - wi /2.;
dist = -smin(-sph, -dist, .15);
sph = distance(cp, vec3(0., 0., -5.)) - 2.;
dist = -smin(sph, -dist, .3);

return dist;


// Anton Roy - antonOTI
// https://www.shadertoy.com/view/WsfXDr
```



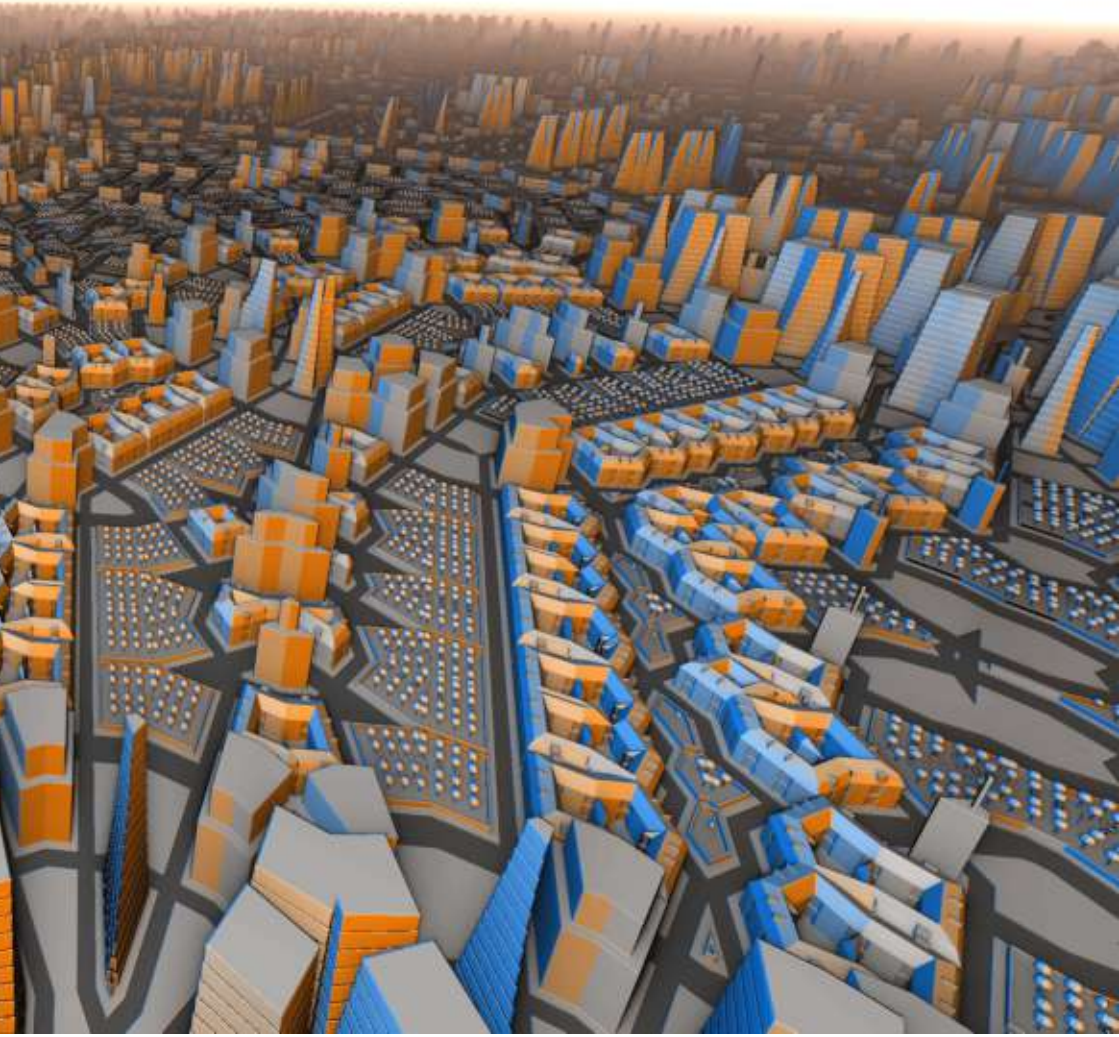
```
void mainImage(out vec4 fragColor, in vec2 fragCoord) {
  vec4 color = vec4(.0, .0, .0, .0);
  vec2 pixel = fragCoord / iResolution.xy * 2.0 - 1.0;
  float time = iTime;
  float rido = sin(10. / pixel.x + cos(time)) * .5 + .5;
  float hSol = 0.1;
  float cx = 2. * pixel.x;
  float cy = 2. * pixel.y + hSol;

  float maskSol = smoothstep(cx, cx+0.4, pixel.x*.3-cy)
    * smoothstep(-(cx+0.4), -cx, pixel.x * .3 - cy);
  float y = 3. * log(abs(pixel.y));
  float x = pixel.x * abs(y - 1.5) * 1.6;

  float h = 0.1;
  float t = h*5.;
  float m = 3.9;
  float a = (m*h/t) * (1. - 2. * step(t/2., mod(x, t)));
  float parquet = step(mod(y + a*x, m/10.), 0.2);

  float smoothCircle = 0.1*cos(iTime)
    + smoothstep(1.5, .0, length(pixel));

  fragColor = mix(
    vec4(1., 0., .2, 1.) * rido,
    vec4(0.85, .77, .6, 1.) * parquet,
    maskSol
  ) * smoothCircle;
}
```

```
// Black Logic by Logzeraw
// https://www.shadertoy.com/view/MtBczD
```



```
// Paris building
vec3 b1 = pos;
b1.xy -= 0.22;
b1 = abs(b1);
rayDist = min(rayDist,
  box(b1, vec3(0., 0., 0.), vec3(.05, .05, .05)));
rayDist = max(rayDist,
  -box(b1, vec3(0., 0., .05), vec3(.02, .02, .05)));

vec3 biz1 = b1;
biz1 -= vec3(.04, .05, .062);
biz1.yz *= rz2(2.5);
rayDist = max(rayDist,
  -box(biz1, vec3(0.), vec3(.02, .02, .02)));

vec3 biz2 = b1;
biz2 -= vec3(.05, .025, .062);
biz2.xz *= rz2(2.5);
rayDist = max(rayDist,
  -box(biz2, vec3(0.), vec3(.02, .022, .02)));

//Garden trees
float treeSize = 1. + .1 * length(sin(pos * 1000.))
  + .5 * length(sin(pos * 100.));

b0.xy = abs(b0.xy);
b0.xy -= 0.02;
b0.xy = abs(b0.xy);
b0 -= vec3(.01, .01, .02);
rayDist = min(rayDist,
  box(b0, vec3(0., 0., -.01), vec3(.001, .001, .005)));
b0 *= treeSize;
b0.xz *= rz2(2.5);
rayDist = min(rayDist, sphere(b0, vec3(0.), .008));
```

```
// Victor Beaupuy - Kushulain
// https://www.shadertoy.com/view/wsXSDn
```



```
float rep = 10.;
p.y -= max(0., abs(p.x) - 10.) * .3;
p.x = (fract(p.x / rep - .5) - .5) * rep;
p += tunnel(p);

vec3 rp = p;
float boxrep = 10.;
rp.z = (fract(rp.z / boxrep - .5) - .5) * boxrep;

vec3 rp2 = p;
float boxrep2 = 1.;
rp2.x=abs(rp2.x) - .4;
rp2.z = (fract(rp2.z / boxrep2 - .5) - .5) * boxrep2;

float b = box(rp + vec3(0, -9, 0), vec3(.6, 10.5, .6));
vec3 rp3 = rp + vec3(0, 1.5, 0);
rp3.xy *= rot(PI * .3);
rp3.yz *= rot(PI * .3);
float b2 = box(rp3, vec3(0.7));
b2 = max(b2, p.y + 1.5);
b = min(b, b2);

float st = stair(p, .1, .4);
float st2 = stair(p + vec3(0, .7, 0), .6, .4);

b = max(b, -st2);

float c = box(rp2 + vec3(0, .3, 0), vec3(.05, .3, .2));
rp2.y = abs(rp2.y + .43) - .1;
c = min(c, box(rp2, vec3(.03, .03, 1.)));

return min(c, min(b, st));
```

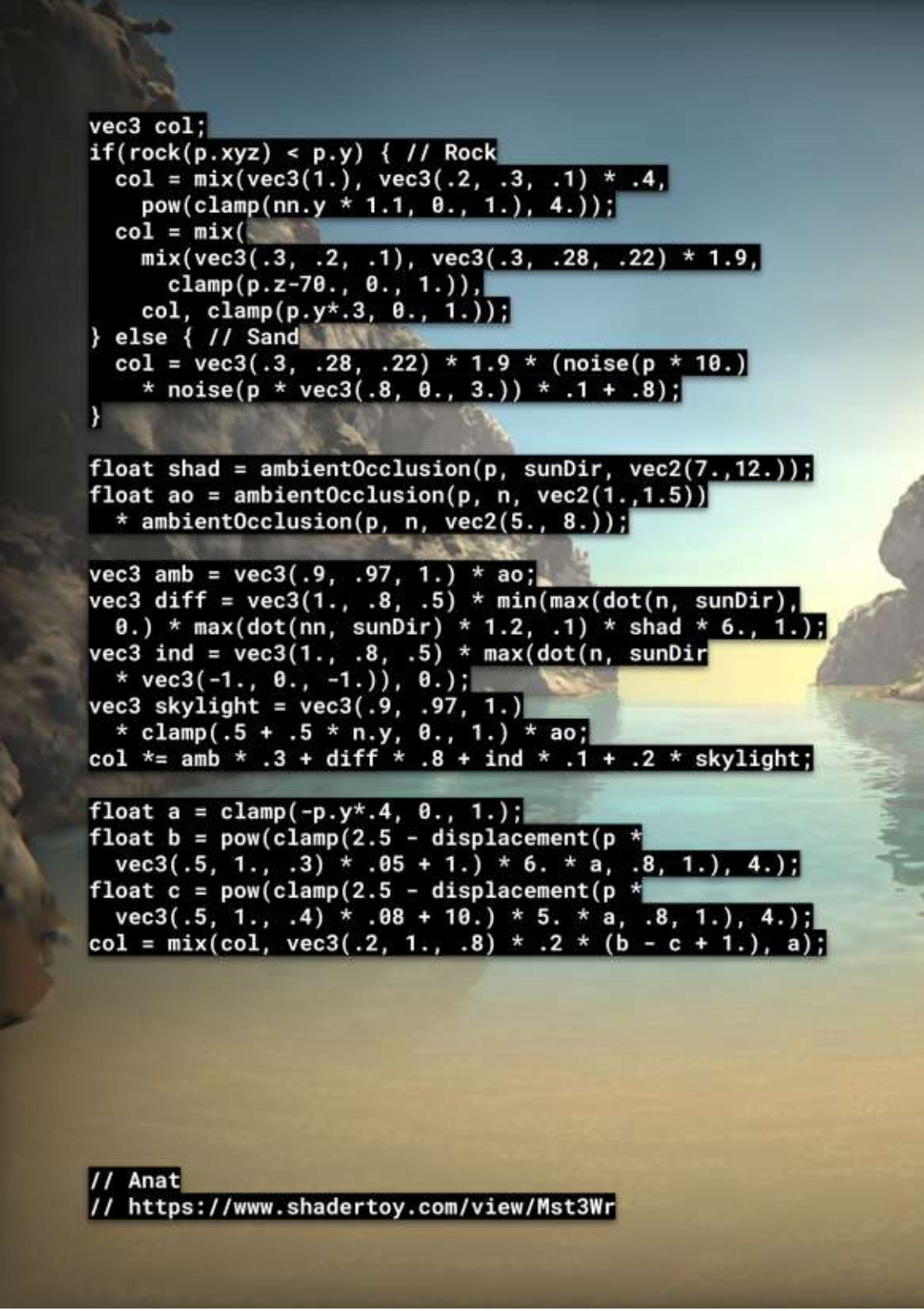
```
// Alice Zanuttini - NuSan
// https://www.shadertoy.com/view/3sBGzV
```

```
// One morning we woke up when the sky was still dark.
// We walked half an hour through the forest,
// to reach the other side of the island,
// where the beach is facing the rising sun.
// The sun was already there, one half over the horizon.
// The sky was on fire.
// We swum in the sea, staring at the rising sun.

float l = dot(n, normalize(vec3(uv.x, -.2, uv.y + .5)));
l = clamp(l, 0., 1.);

vec3 cloudColor = mix(baseSkyColor, darkColor,
  length(uvInit) * 1.7);
cloudColor = mix(cloudColor, sunColor, l);

vec3 skyColor = mix(lightColor, baseSkyColor,
  clamp(uvInit.y * 2., 0., 1.));
skyColor = mix(skyColor, darkColor,
  clamp(uvInit.y * 3. - .8, 0., 1.));
skyColor = mix(skyColor, sunColor,
  clamp(-uvInit.y * 10. + 1.1, 0., 1.));

if(length(uvInit-vec2(0.,.04)) < .03) {
   skyColor += vec3(2.,1.,.8);
}

float cloudMix = clouds * 4. - 8.;
vec3 color = mix(cloudColor, skyColor,
  clamp(cloudMix, 0.,1.));

uvInit.y = abs(uvInit.y);
float islandHeight = texture(iChannel1, uvInit.xx / 2.
  + .67).r / 15. - uvInit.y + .978;
islandHeight += texture(iChannel1, uvInit.xx*2.).r/60.;
islandHeight = clamp(floor(islandHeight),0.,1.);
vec3 landColor = mix(baseSkyColor, darkColor,
  length(uvInit) * 1.5);
color = mix(color, landColor, islandHeight);


// Benjamin Vedrenne - GLKT
// https://www.shadertoy.com/view/4d2BRz
```

```
vec2 fb( vec3 p )
  vec2 h,t=vec2(bo(abs(p)-vec3(2,0,0),vec3(0.3,0.3,10)),5);
  h=vec2(1000,3);
  for(int i=0;i<6;i++)
    h.x=min(h.x,bo(abs(p)-vec3(0,0,0.5*float(i)),
      vec3(2,0.1,0.1)));
    h.x = min(h.x,bo(abs(p)-vec3(2,0,0.5*float(i)),
      vec3(0.2,0.5,0.2)));
  t.x=min(t.x,0.8*(length(p-vec3(0,0,6))-1.7));
  t=(t.x<h.x)?t:h;
  t.x*=0.5;
  return t;
```

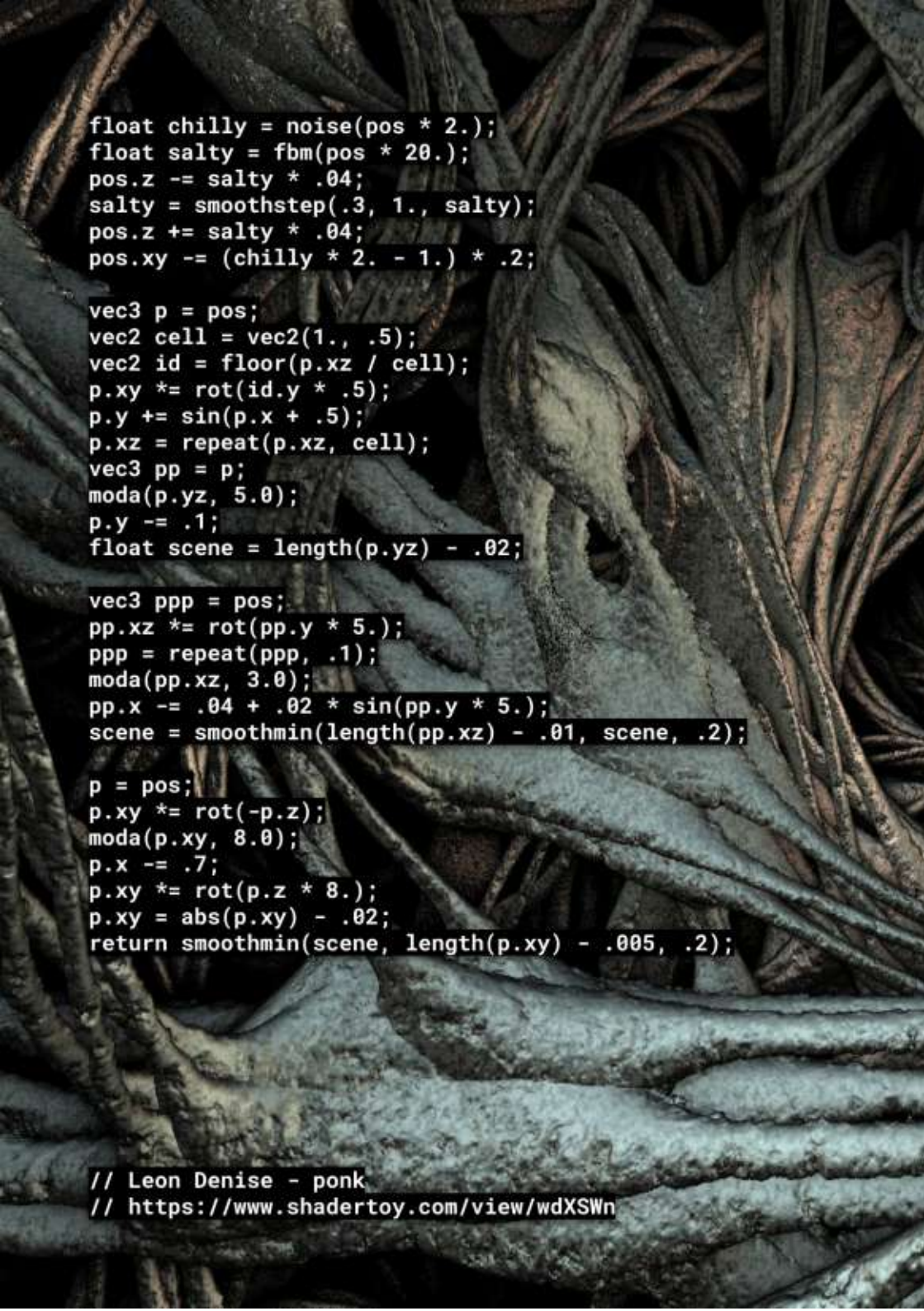
```
vec2 mp( vec3 p )
  p.xy*=r2(sin(p.z*.2)*.5+tt*.5);
  np=p;
  att=length(p-vec3(0,0,sin(tt*2.)*20.))-5.;
  np.z=mod(p.z+tt*10.,15.)-7.5;
  for(int i=0;i<3;i++)
    np=abs(np)-vec3(0.2,0.2+att*0.2,0);
    np.xy*=r2((cos(np.z*.2*float(i))));
  vec2 h,t=fb(np);
  pp=abs(p)-vec3(3.2,1.+sin(p.z*0.2),0)-att*0.2;
  pp.z=mod(pp.z-tt*10.,4.)-2.;
  h=vec2(bo(pp,vec3(0.2+att*0.03)),6);
  h.x=max(-(length(pp)-(0.3+att*0.03)),h.x);
  h.x*=0.7;
  t=(t.x<h.x)?t:h;
  h=vec2(bo(pp,vec3(0.1+att*0.01,0.1+att*0.01,30)),5);
  h.x*=0.8;
  t=(t.x<h.x)?t:h;
  return t;
```

```
// Mangosh Prunier - evvvvil
// https://www.shadertoy.com/view/3s23Dc
```



```
vec3 col;
if(rock(p.xyz) < p.y) { // Rock
  col = mix(vec3(1.), vec3(.2, .3, .1) * .4,
    pow(clamp(nn.y * 1.1, 0., 1.), 4.));
  col = mix(
    mix(vec3(.3, .2, .1), vec3(.3, .28, .22) * 1.9,
      clamp(p.z-70., 0., 1.)),
    col, clamp(p.y*.3, 0., 1.));
} else { // Sand
  col = vec3(.3, .28, .22) * 1.9 * (noise(p * 10.)
    * noise(p * vec3(.8, 0., 3.)) * .1 + .8);
}

float shad = ambientOcclusion(p, sunDir, vec2(7.,12.));
float ao = ambientOcclusion(p, n, vec2(1.,1.5))
  * ambientOcclusion(p, n, vec2(5., 8.));

vec3 amb = vec3(.9, .97, 1.) * ao;
vec3 diff = vec3(1., .8, .5) * min(max(dot(n, sunDir),
  0.) * max(dot(nn, sunDir) * 1.2, .1) * shad * 6., 1.);
vec3 ind = vec3(1., .8, .5) * max(dot(n, sunDir
  * vec3(-1., 0., -1.)), 0.);
vec3 skylight = vec3(.9, .97, 1.)
  * clamp(.5 + .5 * n.y, 0., 1.) * ao;
col *= amb * .3 + diff * .8 + ind * .1 + .2 * skylight;

float a = clamp(-p.y*.4, 0., 1.);
float b = pow(clamp(2.5 - displacement(p *
  vec3(.5, 1., .3) * .05 + 1.) * 6. * a, .8, 1.), 4.);
float c = pow(clamp(2.5 - displacement(p *
  vec3(.5, 1., .4) * .08 + 10.) * 5. * a, .8, 1.), 4.);
col = mix(col, vec3(.2, 1., .8) * .2 * (b - c + 1.), a);
```

```
// Anat
// https://www.shadertoy.com/view/Mst3Wr
```



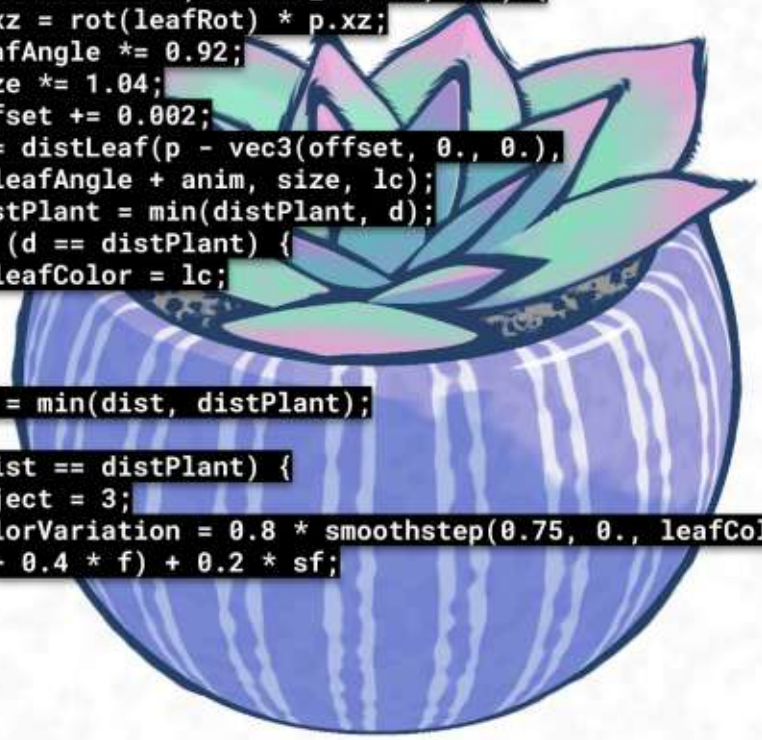
```
float chilly = noise(pos * 2.);
float salty = fbm(pos * 20.);
pos.z -= salty * .04;
salty = smoothstep(.3, 1., salty);
pos.z += salty * .04;
pos.xy -= (chilly * 2. - 1.) * .2;

vec3 p = pos;
vec2 cell = vec2(1., .5);
vec2 id = floor(p.xz / cell);
p.xy *= rot(id.y * .5);
p.y += sin(p.x + .5);
p.xz = repeat(p.xz, cell);
vec3 pp = p;
moda(p.yz, 5.0);
p.y -= .1;
float scene = length(p.yz) - .02;

vec3 ppp = pos;
pp.xz *= rot(pp.y * 5.);
ppp = repeat(ppp, .1);
moda(pp.xz, 3.0);
pp.x -= .04 + .02 * sin(pp.y * 5.);
scene = smoothmin(length(pp.xz) - .01, scene, .2);

p = pos;
p.xy *= rot(-p.z);
moda(p.xy, 8.0);
p.x -= .7;
p.xy *= rot(p.z * 8.);
p.xy = abs(p.xy) - .02;
return smoothmin(scene, length(p.xy) - .005, .2);
```

```
// Leon Denise - ponk
// https://www.shadertoy.com/view/wdXSWn
```

```
apply_transform(p);

p.xy *= r2d(3.14 / 2.);

vec3 q = p;
p.xy *= r2d(3.14 / 4.);
float m = .25;
p.z = mod(p.z - m * .5, m) - m * .5;
p.xy += 4.5;
float b1 = box(p, vec3(.02, 23.5, .1));

p = q;
p -= vec3(0, .3, .6);
float sph1 = length(p) - .3;
p -= vec3(-.1, .4, 1.5);
float sph2 = length(p) - .05;

p = q;
mo(p.xz, vec2(2.8));
mo(p.xy, vec2(1.2));

mo(p.xz, vec2(2.03, .3));
p.xy *= r2d(3.14 / 2.2);

mo(p.xz, vec2(.5, .4));
amod2(p.xz, 8.);
float d = od(p, .7);
d = cmin(d, sc(p, .03), .3);
d = max(d, box(p, vec3(4)));

return min(min(min(d, sph2), sph1), b1);
```

```
// Théotime Calandra - lsdlive
// https://www.shadertoy.com/view/tsfSWr
```

```
p = pos;
p.y -= 0.2;
float distPlant = 100.;
float anim = 0.05 * (0.5 + 0.5 * sin(5. * iTime));
float leafAngle = 1.2;
float offset = 0.01;
float size = 0.2;
float leafRot = 2. * PI / PHI;
float leafColor, lc, d;

for(float i = 0.; i < NB_LEAVES; i++) {
  p.xz = rot(leafRot) * p.xz;
  leafAngle *= 0.92;
  size *= 1.04;
  offset += 0.002;
  d = distLeaf(p - vec3(offset, 0., 0.),
    leafAngle + anim, size, lc);
  distPlant = min(distPlant, d);
  if (d == distPlant) {
    leafColor = lc;
  }
}

dist = min(dist, distPlant);

if(dist == distPlant) {
  object = 3;
  colorVariation = 0.8 * smoothstep(0.75, 0., leafColor
    + 0.4 * f) + 0.2 * sf;
}
```



```
// Hélène Legrand - lnae
// https://www.shadertoy.com/view/4ltyD4
```

```
vec3 n = get_norm(p);
float fre = pow(clamp(1. - dot(n, -rd), 0., 1.), 3.);
vec3 keypos = vec3(-8., 10., -10.);
vec3 rimpos = vec3(5., -2., 5.);
vec3 specpos = vec3(-3.,6., -3.);
vec3 global_lighting = mix(vec3(.2, .1,0.),
  vec3(.8, .8, .5), dirlight(n, keypos));
vec3 rim_lighting = vec3(.1, .5, .8)
  * dirlight(n, rimpos);

if (mat == 0) {
  vec3 albedo = vec3(rand(t_id + 2.), 1.,
    rand(t_id + 4.)) * 0.5;
  col = albedo;
  col += global_lighting + rim_lighting;
  col += spec_light(rd, n, specpos, 2.);
  col /= 3.;
} else if (mat == 1) {
  vec3 albedo = vec3(.8, .8, 1.);
  col = albedo;
  col += global_lighting + rim_lighting;
  col /= 2.;
  col += fre;
} else if (mat == 2) {
  vec3 albedo = vec3(.8, .7, 0.);
  col = albedo;
  col += global_lighting + rim_lighting;
  col += spec_light(rd, n, keypos, 16.)
      * vec3(1., .6, .5);
  col /= 3.;
  col += fre;
} else if (mat == 3) {
  col = global_lighting;
}



// Flopine
// https://www.shadertoy.com/view/wslXR7
```

```
float l = length(p.xz);
float at = atan(p.z, p.x);
float d = INF;

// Petals
for (int ri = 0; ri < 15; ++ri) {
  float RATIO = float(ri) / 15.;
  float ANGLE = random(float(ri)) * TAU;
  float SPANANGLE = 2.0 - RATIO * 1.;
  float RADIUS = .1 + .6 * RATIO;
  float HEIGHT = mix(.8, 1., sqrt(RATIO));
  float OPENING = mix(-.1, .5, RATIO);
  float a = amod(at - ANGLE, TAU);
  float dr = min(d, (abs(a) - SPANANGLE) * l);
  float r = atan(p.y * 5.) * 4. / TAU * RADIUS
    + smoothstep(HEIGHT - .5, HEIGHT + .5, p.y)
    * OPENING * RADIUS;
  dr = max(dr, -length(p.xz) + r);
  dr = max(dr, length(p.xz) - r - .02);
  dr = max(dr, max(p.y - HEIGHT + pow(a, 2.) * .2, 0.));
  d = min(d, dr);
}

// Leaves
for (int li = 0; li < 5; ++li) {
  float RATIO = float(li) / 5.;
  vec3 q = p;
  q.x -  = pow(q.y + .2, 2.) * .2;
  q.y += .4 + float(li) * .3 + random(float(li)) * .05;
  q.yz *= rz2(-(p.y + .2) * .2);
  q.xz *= rz2(fract(sin(float(li) * 13. + 45.)) * TAU);
  d = smin(d, leaf(q, .5 + RATIO * .3), .08);
}
m = mmin(mleaves, M(d, GREEN));




// Jonathan Giroux - Koltes
// https://www.shadertoy.com/view/Md2yDd
```

Cookie Collective gathers digital artists involved in real-time creation. This covers video games, art installations, video mapping, demoscene, live coding, etc.

We organize the Cookie Demoparty, a digital art festival which brings together various communities and features demos, experimental video games, machinimas, and concerts.

And regular live coding parties, in which artists train and create visuals and sounds in front of the audience.0

www.cookie.paris