WEEK 8
AMBIENT: RANDOM, NOISE, MATHS

Pamukkale Abstract by Dimitar Karanikolov

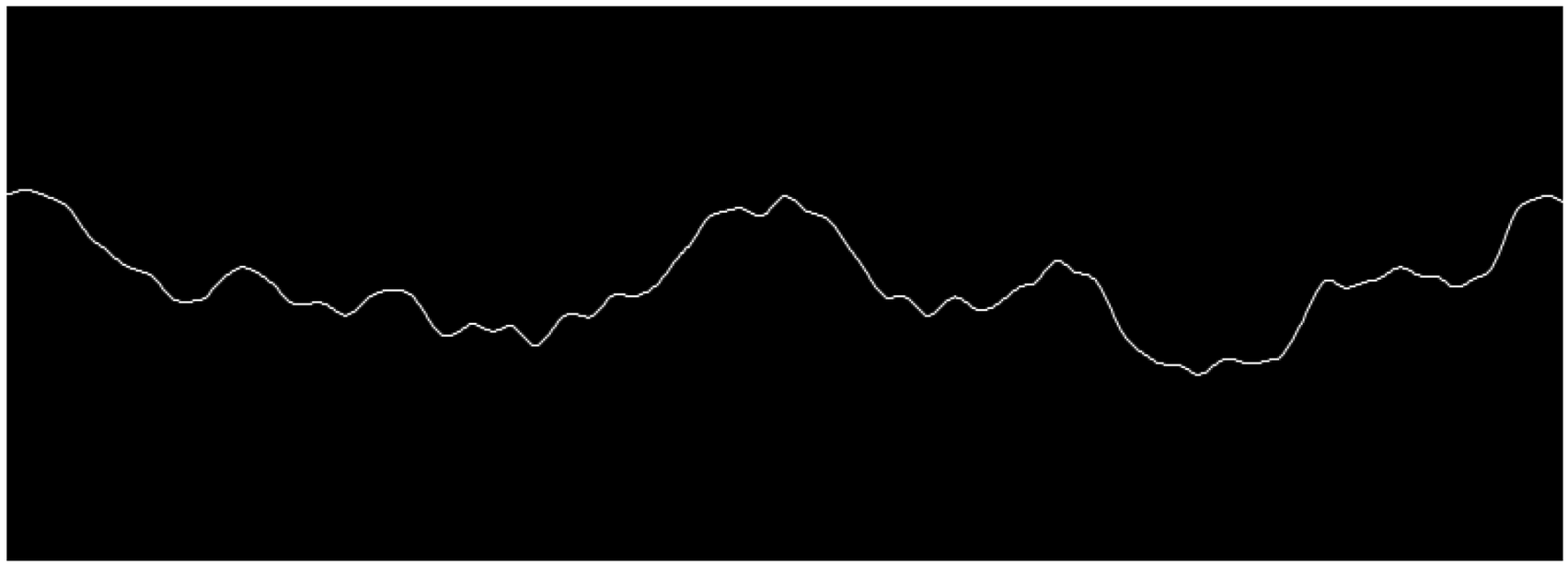
Shapes by connecting vertices

- vertex() is used within the beginShape() and endShape() functions to specify the vertex coordinates for points, lines, triangles, quads, and polygons
- beginShape() begins recording vertices for a shape
- endShape() stops recording
- Example:

```
beginShape();
vertex(120, 80);
vertex(340, 80);
vertex(340, 300);
vertex(120, 300);
endShape();
```

Re-doing Wk7 Example 7c

```
float tt = 50; // change with time

void setup() {
  size(800, 600);
  noFill();
  stroke(255);
}

void draw() {
  background(0);
  float tx = 0; // change with horizontal position
  beginShape();
  for (int x = 0; x < width - 1; x++) {
    float y = 150 + noise(tx, tt) * 200;
    vertex(x, y);
    tx += 0.01;
  }
  endShape();
  tt += 0.003;
}
```



shape-related

time-related

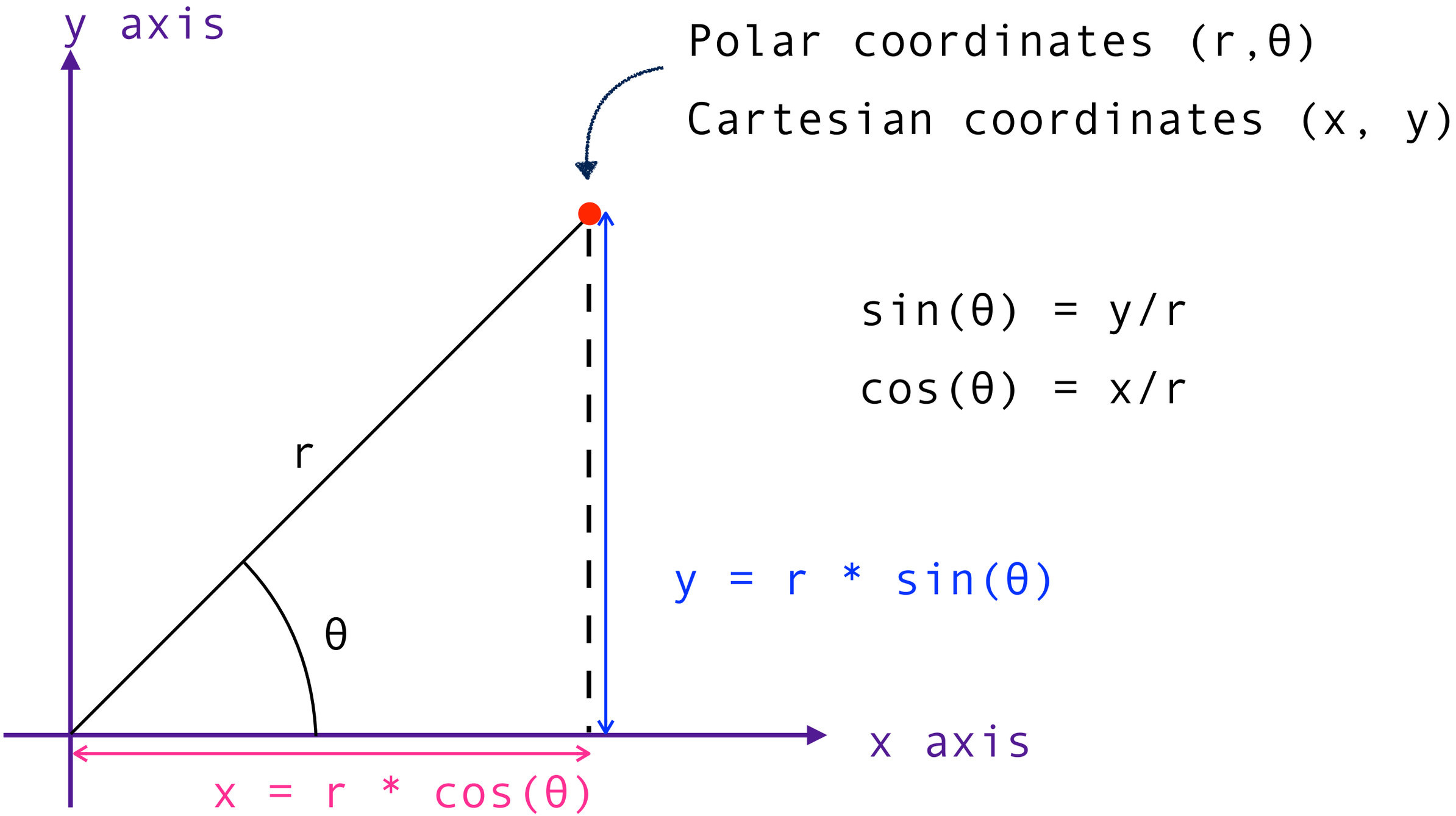
Example 1a

```
int numCurves = 50;

float tt = 50; // change with time

void setup() {
  //size(800, 600);
  fullScreen();
  noFill();
  strokeWeight(2);
  stroke(255);
  noCursor();
}

void draw() {
  background(0);
  float ty = 100; // change with vertical position
  for (int i = 0; i < numCurves; i++) {
    float tx = 0; // change with horizontal position
    beginShape();
    float alpha = map(i, 0, numCurves, 50, 255);
    stroke(255, alpha);
    for (int x = 0; x < width -1; x+=10){
      float y = i*20 + noise(tx, ty, tt) * 400 - 50;
      vertex(x, y);
      tx += 0.02;
    }
    endShape();
    ty += 0.05;
  }
  tt += 0.001;
}
```

3D noise function!

Example 1b

Cartesian vs Polar Coordinates

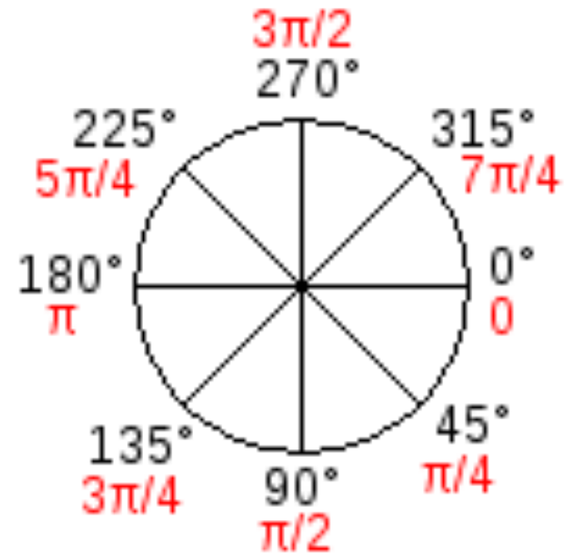


Trigonometry in Processing

Functions:

acos()
asin()
atan2()
atan()
cos()
degrees()
radians()
sin()
tan()

sin(angle), cos(angle)
- angle: in radians

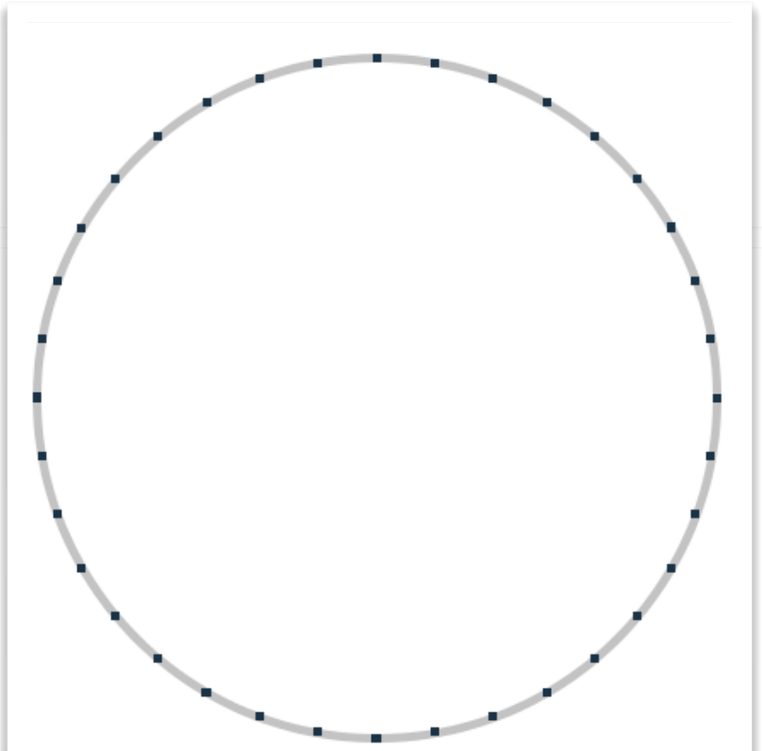


radians(degrees): convert a degree measurement to its corresponding value in radians
- radians = 2*PI*(degrees/360)

Rotational Drawing

Example 2a: draw a circle using trigonometry

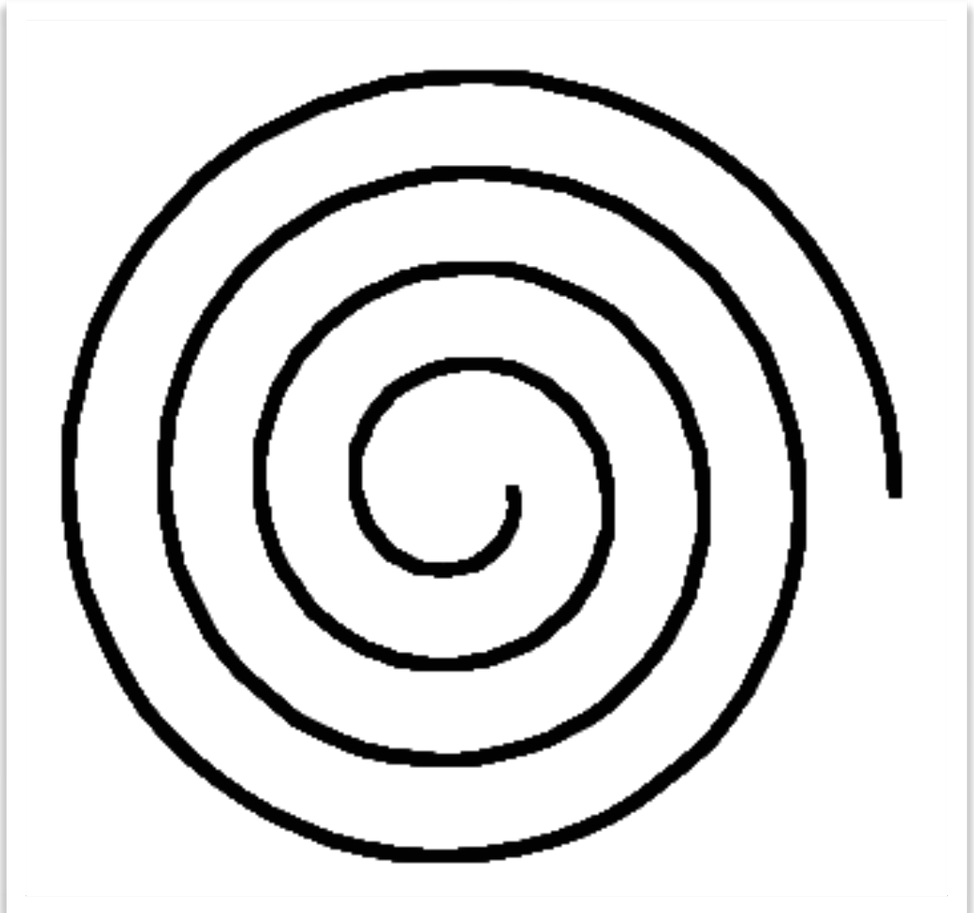
```
float x, y, radius = 200, centX, centY;

size(500, 500);
background(255);
centX = width / 2;
centY = height / 2;
strokeWeight(5);
stroke(0, 60);
noFill();
ellipse(centX, centY, radius*2, radius*2);
stroke(0);
for (float ang = 0; ang <= 360; ang += 10) {
  float rad = radians(ang);
  x = centX + (radius * cos(rad));
  y = centY + (radius * sin(rad));
  point(x, y);
}
```



Spiral

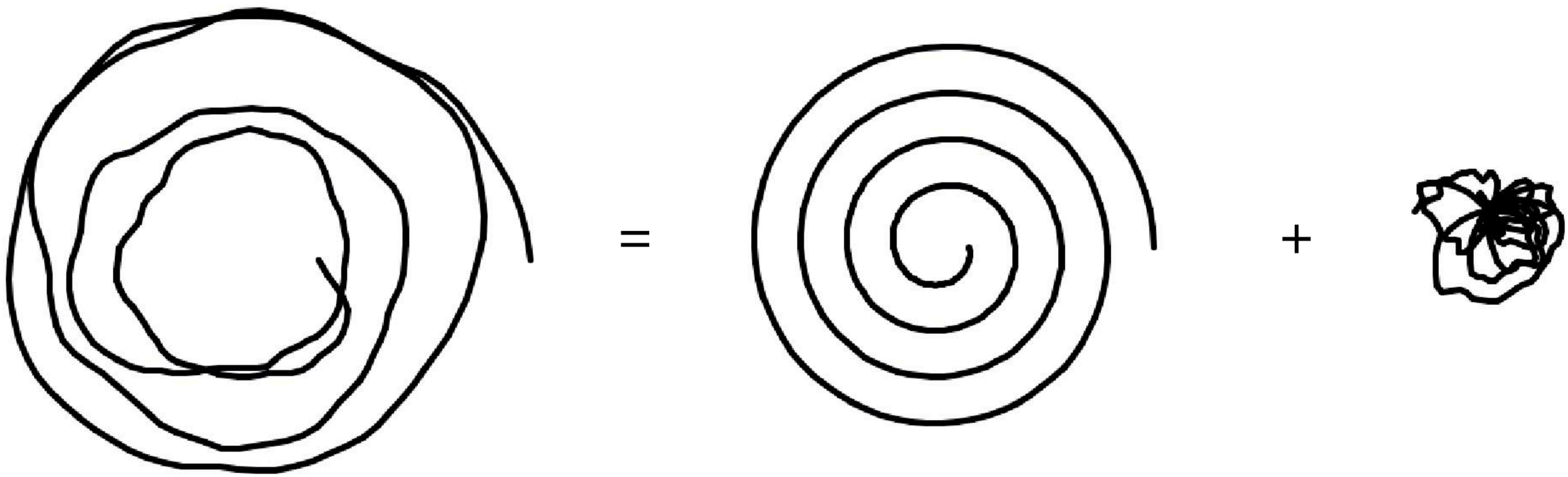
Example 2b: turn a circle into a spiral

```
radius = 20;
float lastx = -999, lasty = -999;

for (float ang = 0; ang <= 360 * 4; ang += 5) { // loop for four times
  float rad = radians(ang);

  radius += 0.5;

  x = centX + (radius * cos(rad));
  y = centY + (radius * sin(rad));

  if (lastx > 0 )
    line(x, y, lastx, lasty);

  lastx = x;
  lasty = y;
}
```



Example 2c: Noisy Spiral

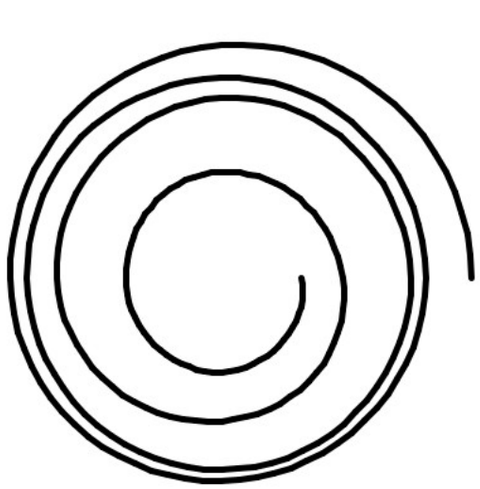
```
float r_noise    =    radius    +    noise(t) * 100;
```



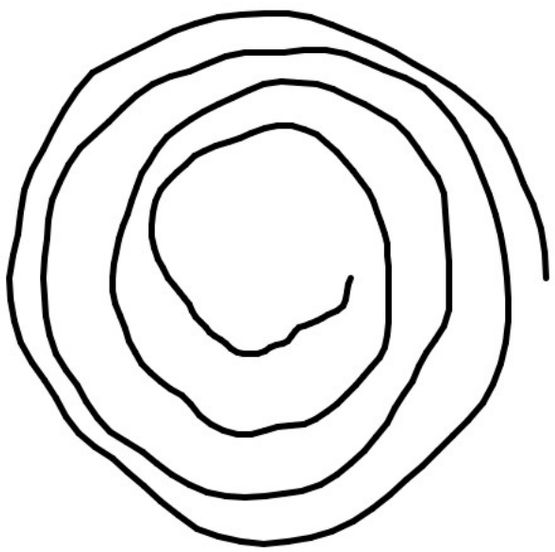
Example 2c: Noisy Spiral

```
radius = 20;
t = random(10);
for (float ang = 0; ang <= 360 * 4; ang += 5) { // loop for four times
  float rad = radians(ang);
  radius += 0.5;

  float r_noise = radius + noise(t) * 100;

  x = centX + (r_noise * cos(rad));
  y = centY + (r_noise * sin(rad));

  if (lastx > 0 ) // lastx and lasty have been intliaized
    line(x, y, lastx, lasty);

  lastx = x;
  lasty = y;

  t += 0.05;
}
```

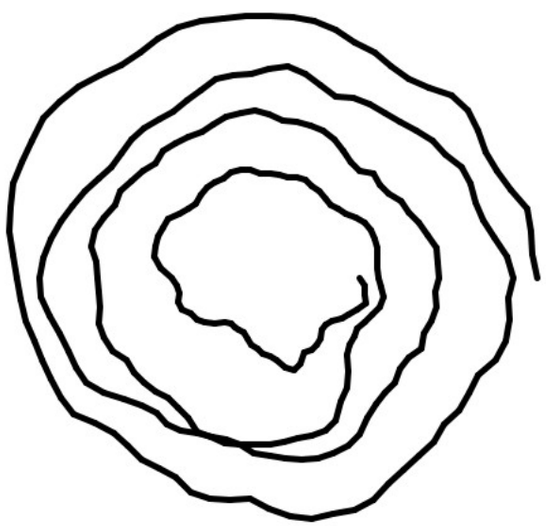
what if t=0 for every frame?



t += 0.01;



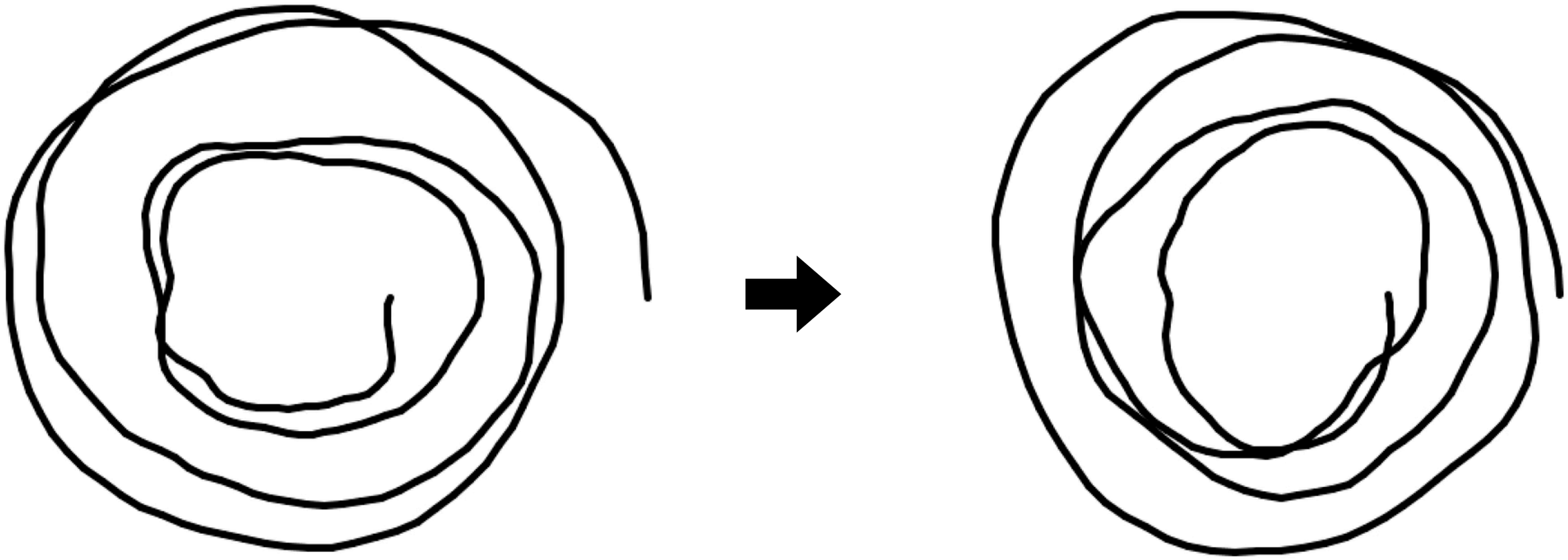
t += 0.05;



t += 0.09;

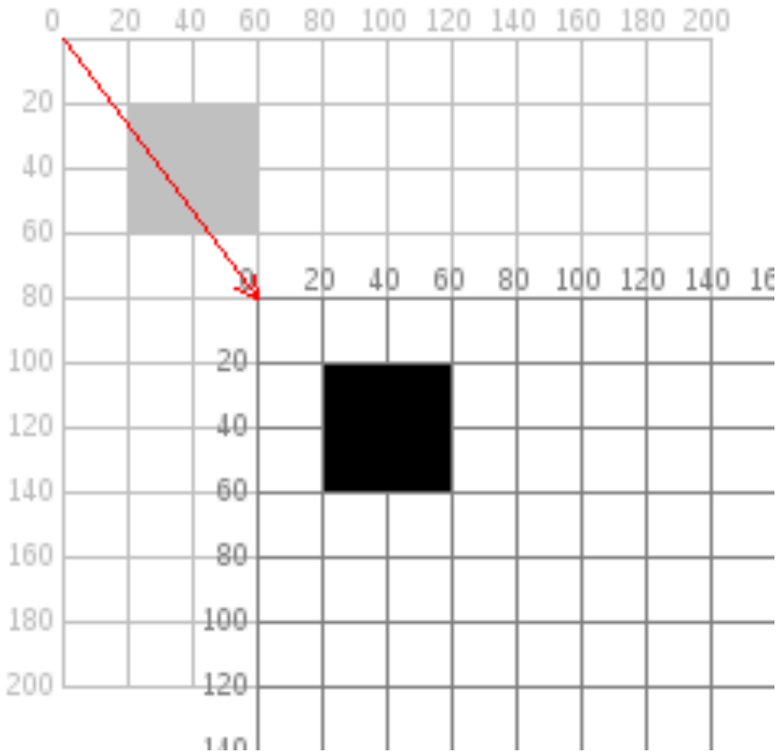
Exercise 1

- Modify Example 2c to make the noisy spiral to perform continuous motion
- You may refer to Example 1a

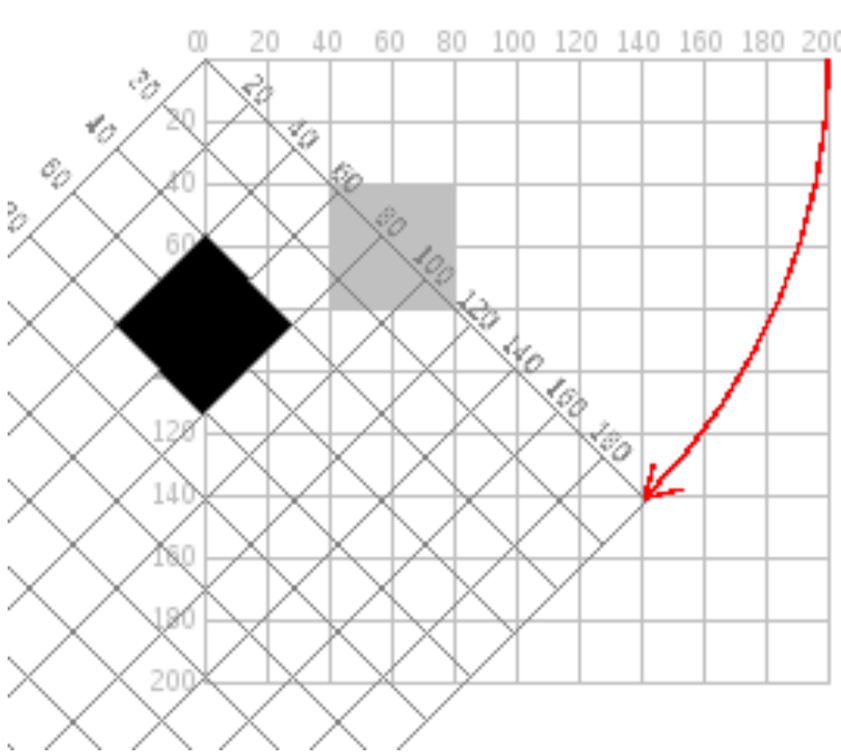


2D Transformations

Transform the coordinate system




```
translate(60, 80);
rect(20, 20, 40, 40);
```




```
rotate(radians(45));
rect(40, 40, 40, 40);
```

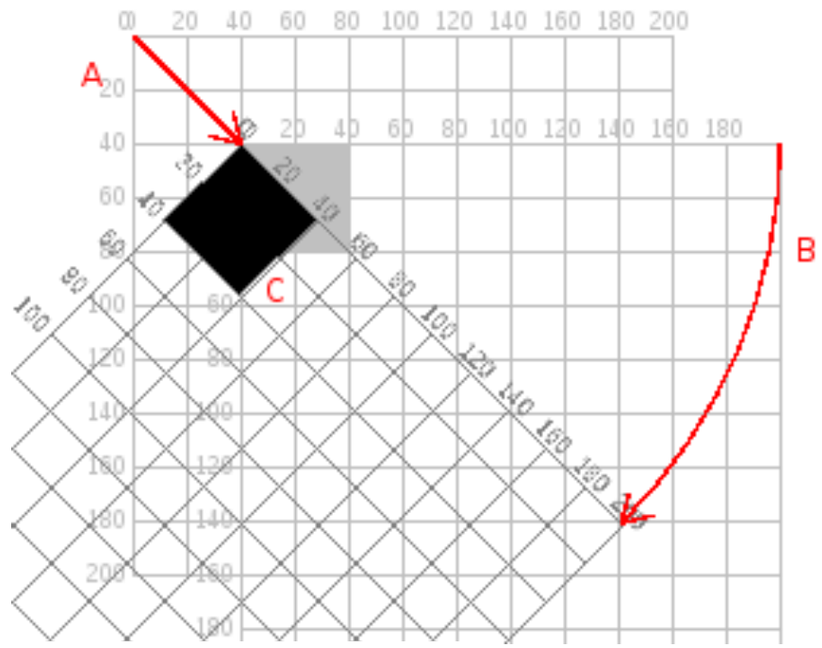
Transform

applyMatrix()
popMatrix()
printMatrix()
pushMatrix()
resetMatrix()
rotate()
rotateX()
rotateY()
rotateZ()
scale()
shearX()
shearY()
translate()

2D Transformations

- Transformations are cumulative
- Transformations apply to everything that happens after
- Transformations are reset at the beginning of draw() loop

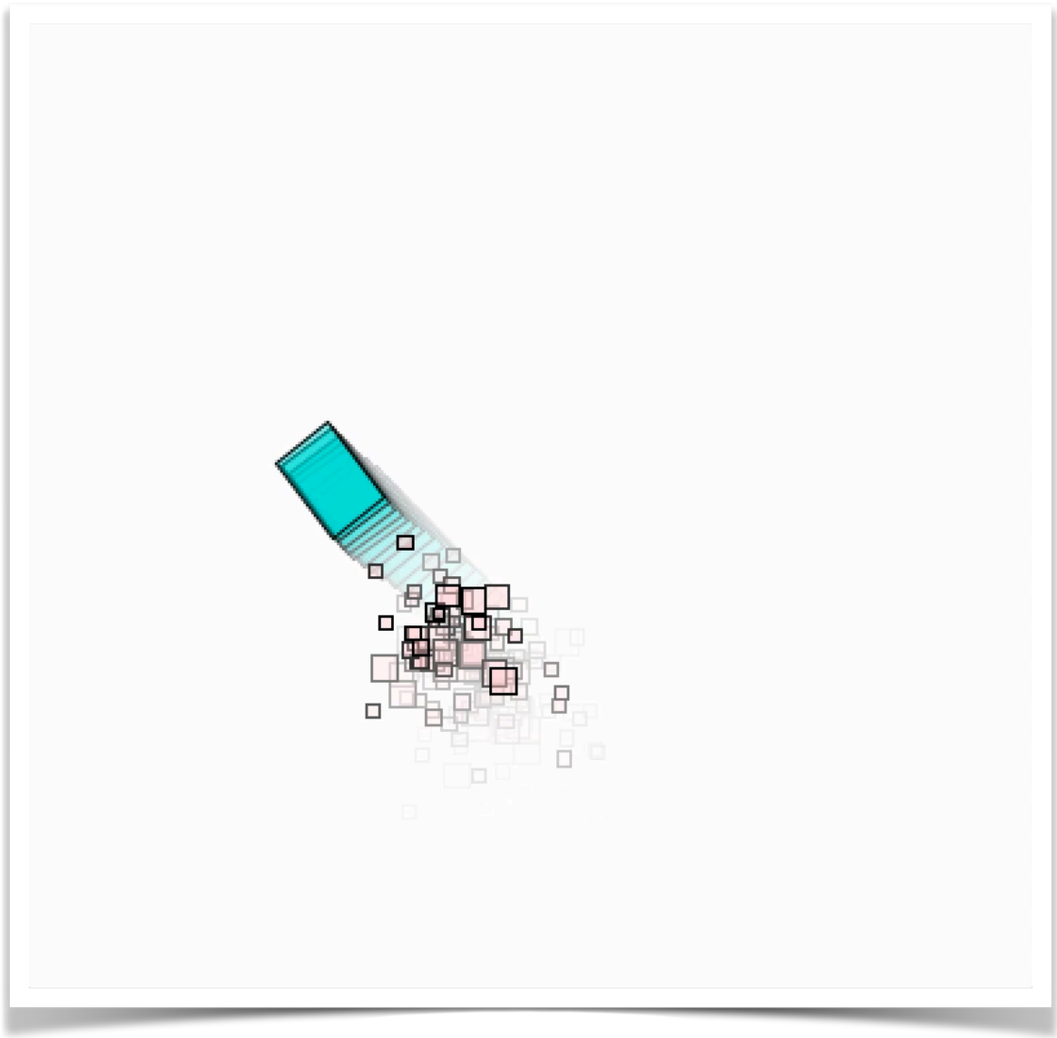
```
// move the origin to the pivot point
translate(40, 40);
// then pivot the grid
rotate(radians(45));
// and draw the square at the origin
fill(0);
rect(0, 0, 40, 40);
```



Example 3

Example 4

```
// followers
for (int i = 0; i < count; i++) {
  ex[i] = lerp(x, ex[i], .95);
  ey[i] = lerp(y, ey[i], .95);
  fill((i+1)*100+100, 200, 200, 100);
  rect(ex[i]+i*random(-1, 1)*5, ey[i]+i*random(-1, 1)*5, es[i], es[i]);
}

// target
translate(x, y);
rotate(r);
fill(0, 200, 200, 100);
rect(0, 0, w, h);

t = t + 0.01;
```



Transformation Matrix

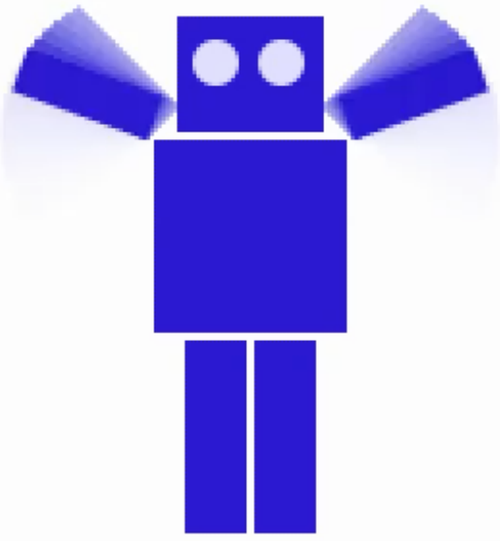
Matrix stack:

pushMatrix() - saves the current coordinate system and

popMatrix() - restores the prior coordinate system.

Transformations between pushMatrix() and popMatrix() will not apply to anything that happens after popMatrix()

```
void drawLeftArm(){
  pushMatrix();
  translate(12, 32);
  rotate(radians(armAngle));
  rect(-12, 0, 12, 37); // left arm
  popMatrix();
}

void drawRightArm(){
  pushMatrix();
  translate(66, 32);
  rotate(radians(-armAngle));
  rect(0, 0, 12, 37); // right arm
  popMatrix();
}
```



Example 5

Example 6 random walk with rotation

```
void update() {
  pushMatrix();
  translate(x, y);
  rotate(a);
  fill(c, 50);
  text(s, 0, 0);
  popMatrix();

  tx += 0.01;
  ty += 0.01;
  ta += 0.001;

  x += noise(tx)*4-2;
  y += noise(ty)*4-2;
  a += noise(ta)*0.1-0.05;

  if (x<0)
    x += width;
  else if (x>width)
    x -= width;
  if (y<0)
    y += height;
  else if (y>height)
    y -= height;
}
```

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

beginShape();
for (int deg = 0; deg <= 360; deg+=1) {
  float angle = radians(deg);
  float x = cos(angle);
  float y = sin(angle);
  float n = noise(x+frameCount*0.005, y+frameCount*0.005);
  x *= n*radius;
  y *= n*radius;
  vertex(x, y);
}
endShape();
radius+=5;
if (radius>width*2) {
  radius = 5;
  background(0);
}
```

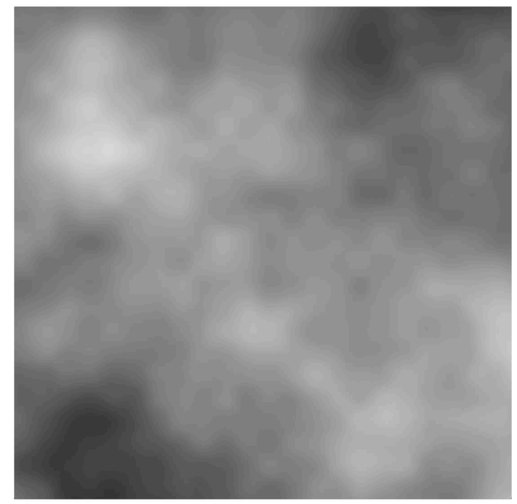
“looped” noise using polar coordinates

Example 7

Example 8: 2D Noise Visualisation

```
xstart += 0.01;
ystart += 0.01;

ynoise = ystart;
for (int y = 0; y <= height; y++) {
  ynoise += 0.01;
  xnoise = xstart;
  for (int x = 0; x <= width; x++) {
    xnoise += 0.01;
    set(x, y, color(255*noise(xnoise, ynoise)));
  }
}
```

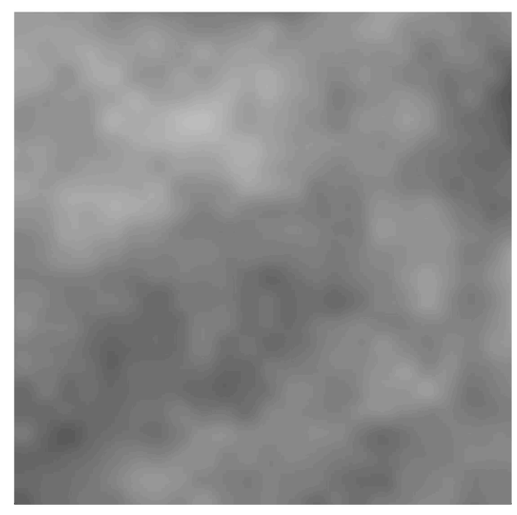
shifting the starting point



Example 8a

```
ynoise = ystart;

tnoise +=0.005;

for (int y = 0; y <= height; y+=1) {
  ynoise += 0.01;
  xnoise = xstart;
  for (int x = 0; x <= width; x+=1) {
    xnoise += 0.01;
    set(x, y, color(255*noise(xnoise, ynoise, tnoise)));
  }
}
```

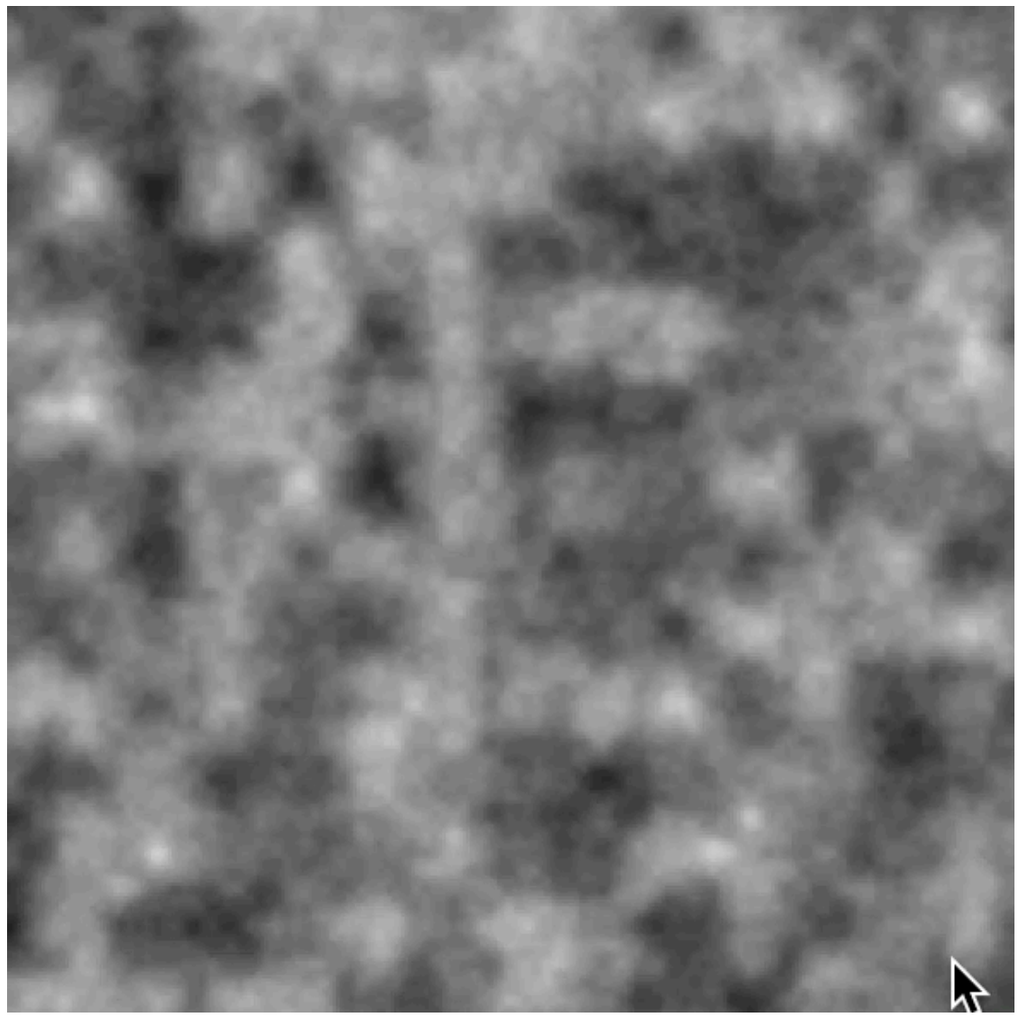
time-related noise parameter



Example 8b

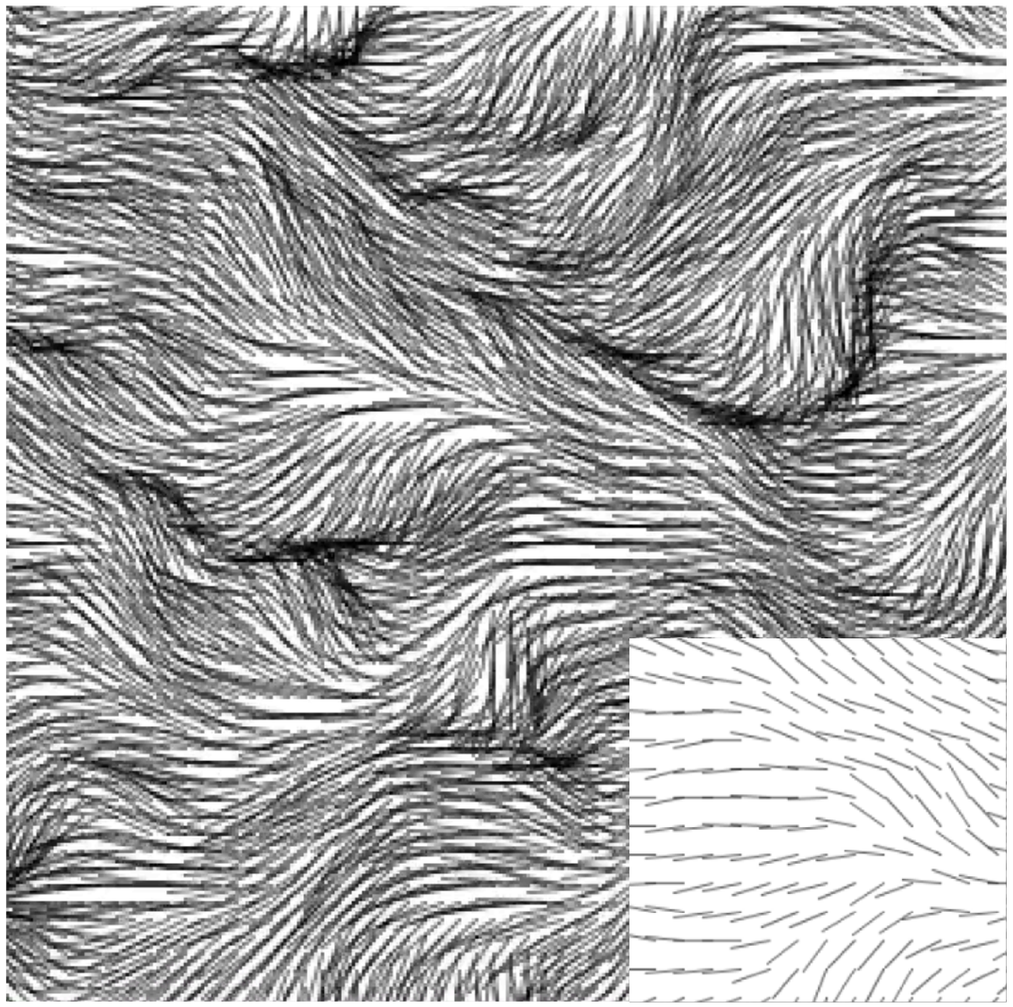
Example 8c: noise scale

```
// control spatial increment with mouseX
dxy = map(mouseX, 0, width, 0.001, 0.05);

// control temporal increment with mouseY
dt = map(mouseY, 0, height, 0.001, 0.05);

ynoise = ystart;

tnoise += dt;

for (int y = 0; y <= height; y+=1) {
  ynoise += dxy;
  xnoise = xstart;
  for (int x = 0; x <= width; x+=1) {
    xnoise += dxy;
    set(x, y, color(255*noise(xnoise, ynoise, tnoise)));
  }
}
```

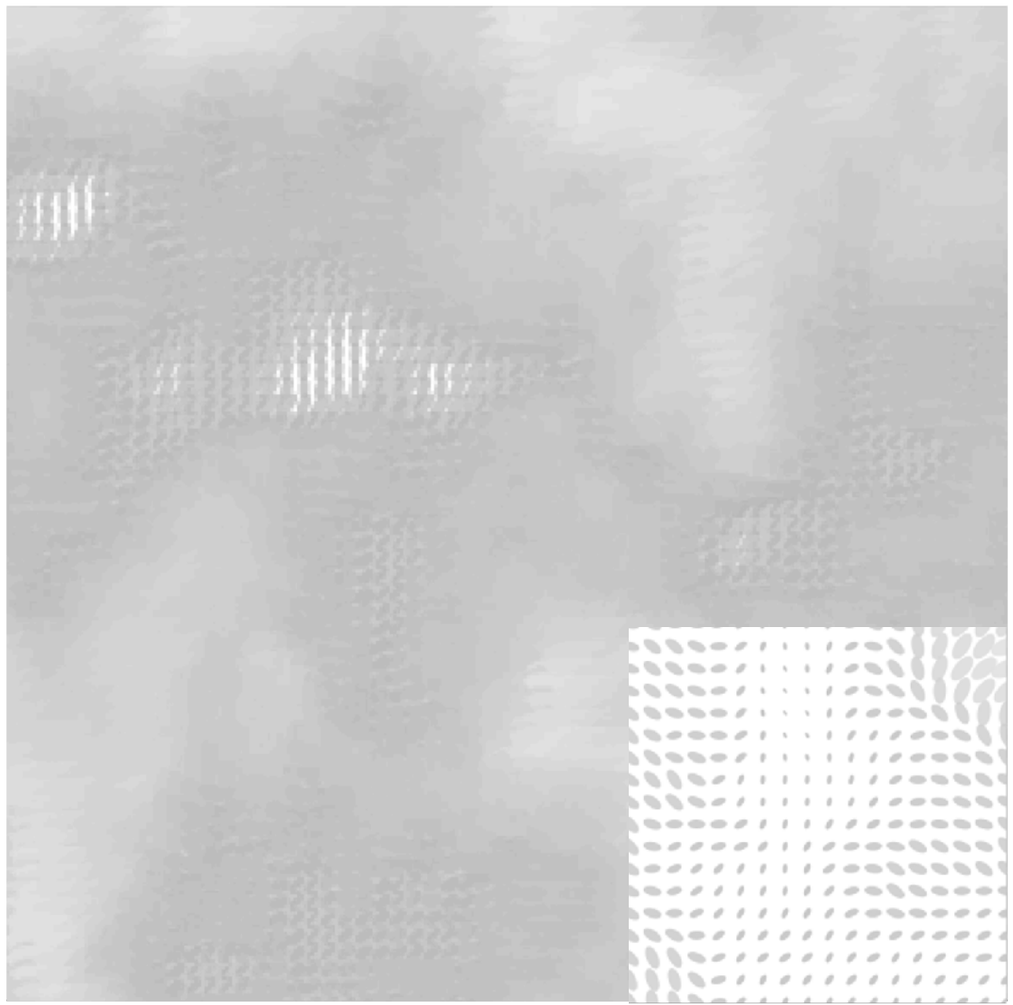


Exercise 2

Modify Example 8b and apply coordinate transformation to achieve different noise visualisation:
i. line() ii.ellipse()
- create user-defined functions
- increase space between points



Visualised using rotating lines



Fluffy clouds