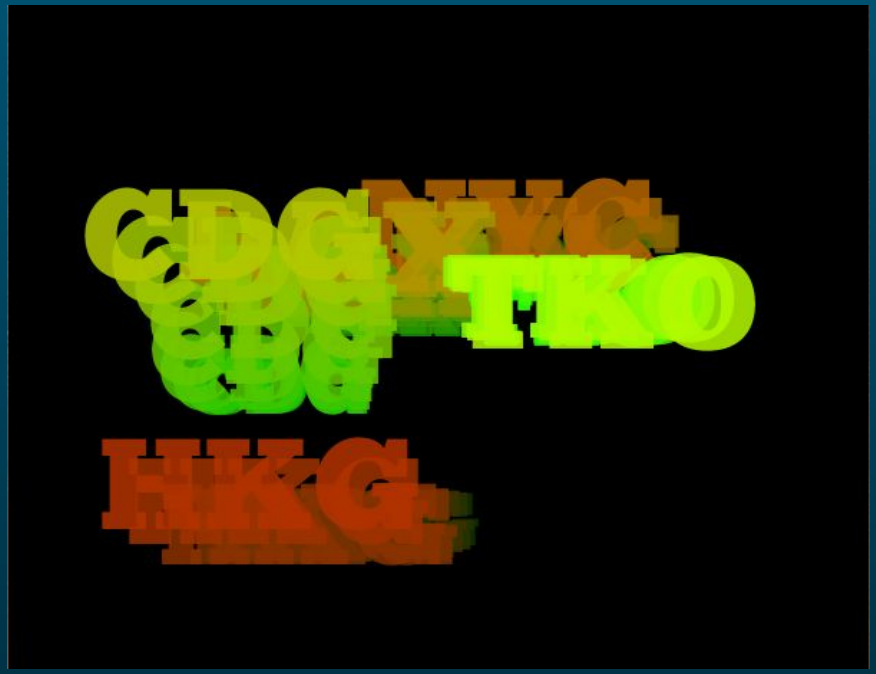
Week 07
Art of Noise

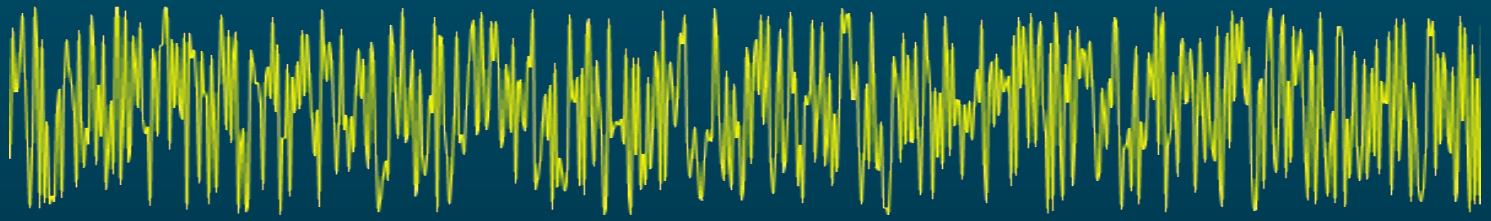
by Mike Wong. School of Creative Media

TypoArt with `random()`

TypoArt with `noise()`



Quick Review of `random()`



Quick Review of random()

`random(<high>);`
`random(<low>, <high>);`
accepts one or two number as parameter(s). The parameters define the range of the output random number.

```
// Outputs from 0 to 2.5 (not including 2.5 itself)
random(2.5);

// Outputs from -2.5 to 3.0 (not including 3.0 itself)
random(-2.5, 3.0);
```

Pseudo Random Number Sequence

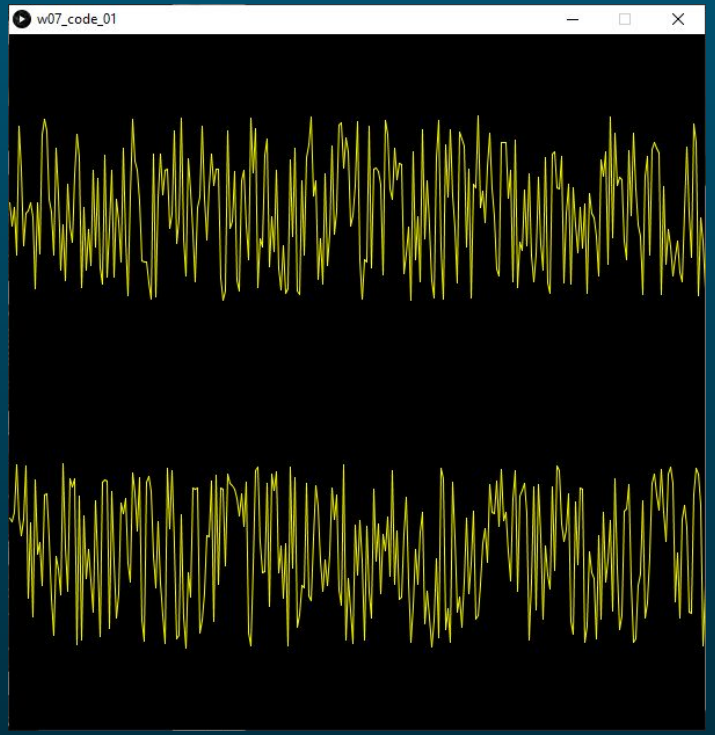
It is common to call `random()` repeatedly for creative coding purpose, and the artists might want to use the same set of random numbers. A sequence of reproducible random number is called a Pseudo Random Number Sequence. We may use the function `randomSeed(int s)` to control the seeding of a random number sequence generation. It must be invoked before calling the function `random()`.

```
// Initializes the PRNG with randomSeed()
randomSeed(10);
random(0,10);
```

Example 1 - use of randomSeed()

w07_code_01

```
void setup() {
  size(600, 600);
  background(0);
  stroke(255, 255, 0);

  //  Upper Plot
  translate(0,150);
  plotRandom(1, -80, 80);
  //  Lower Plot
  translate(0,300);
  plotRandom(1, -80, 80);
}

void plotRandom(int s, int b, int e) {
//  randomSeed(s);
  float lastY = random(b,e);
  for (int x=0; x < width; x+=2) {
    float nowY = random(b,e);
    line(x, lastY, x+2, nowY);
    lastY = nowY;
  }
}
```



Example 2 - Static TypoArt

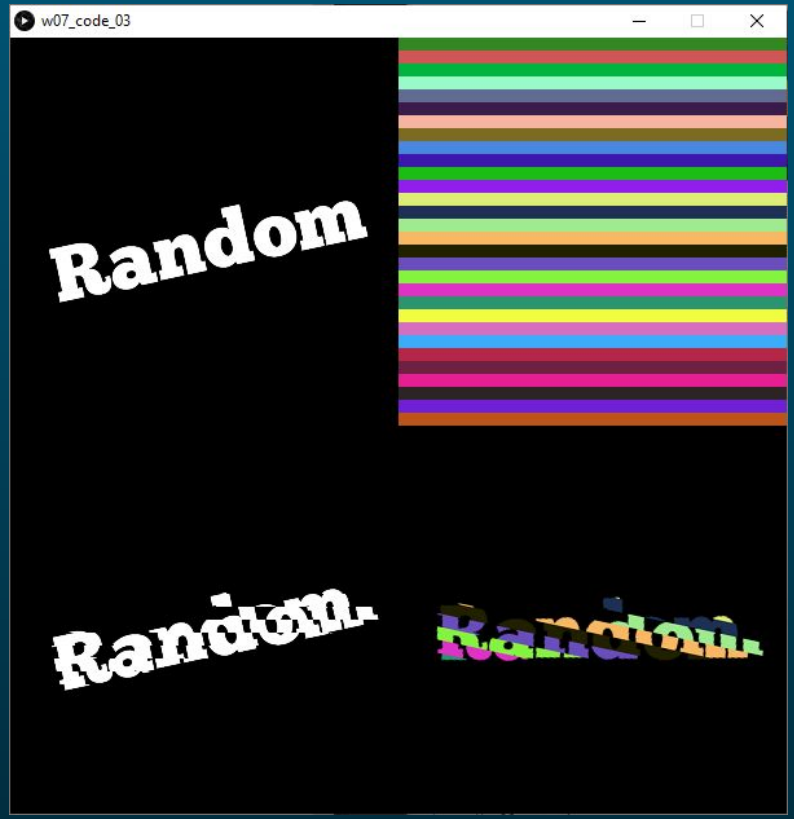
w07_code_02

```
int numSlices = 50;
int offset = 5;

void setup() {
  size(600, 600);
  PFont myFont = createFont("ChunkFive-Regular.otf", 32);
  textAlign(CENTER, CENTER);
  textFont(myFont);
  textSize(60);

  background(0);
  float rot = 45;

  //  Draw the text (Uppest Region)
  pushMatrix();
  translate(150,150);
  rotate(radians(rot));
  text("Random", 0, 0);
  popMatrix();

  //  Sliced copy with random offset (Middle region)
  int sliceH = 300/numSlices;
  for (int i = 0; i < numSlices; i++) {
    int y = round(map(i, 0,numSlices, 0, 300));
    copy(0,y, 300,sliceH, round(random(-offset,offset)),y+300, 300,sliceH);
  }

  //  Display (Bottom)
  PImage buffer = get(0,300, 300,300);
  buffer.mask(buffer);

  translate(450,150);
  rotate(radians(-rot));
  imageMode(CENTER);
  tint(255,255,0);
  image(buffer,0,0);
}
```

Example 3 - Animated TypoArt

w07_code_03

```
int numSlices = 30;
int offset = 10;

void setup() {
  size(600, 600);
  PFont myFont = createFont("ChunkFive-Regular.otf", 32);
  textAlign(CENTER, CENTER);
  textFont(myFont);
  textSize(60);
  noStroke();
}

void draw() {
  background(0);
  float rot = radians(frameCount % 360);  // Animated #1

  //  Draw the text (Upper Left)
  pushMatrix();
  translate(150,150);
  rotate(rot);
  fill(255);
  text("Random", 0, 0);
  popMatrix();

  //  Sliced copy with random offset as mask (Lower Left)
  int sliceH = 300/numSlices;
  for (int i = 0; i < numSlices; i++) {
    int y = round(map(i, 0,numSlices, 0, 300));
    copy(0,y, 300,sliceH, round(random(-offset,offset)),y+300, 300,sliceH);
  }

  //  Rainbow
  for (int i = 0; i < numSlices; i++) {
    int y = round(map(i, 0,numSlices, 0, 300));
    fill(random(255), random(255), random(255));
    rect(300,y, 300, sliceH);
  }

  //  Display (Bottom)
  PImage rgbBuffer  = get(300,0, 300,300); // cap. upper Right
  PImage maskBuffer = get(0,300, 300,300); // cap. lower Left
  rgbBuffer.mask(maskBuffer);

  translate(450,450);
  rotate(-rot);
  imageMode(CENTER);
  image(rgbBuffer,0,0);
}
```



noise(): A friend of random()

The numbers generated by successive calls to random() are almost guaranteed to be un-correlated and independent. When random() is used for animation, the motion looks jumpy.

It is quite often that we need a sequence of numbers to animate things such that they have smooth and natural motion. In short, we need something random but smooth.

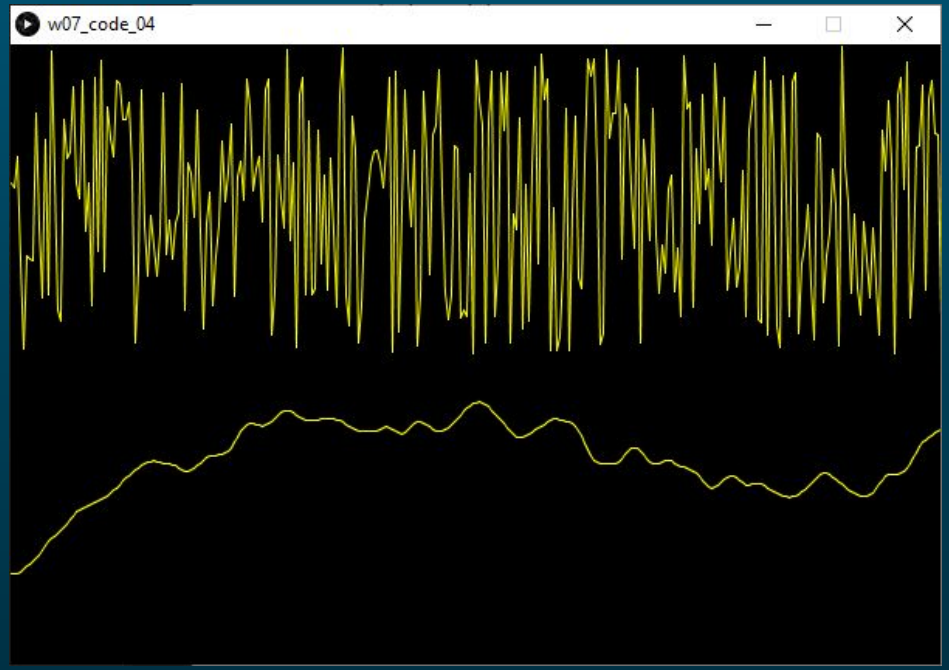
The function noise() serves exactly this purpose. Processing's noise function uses the classic noise function introduced by **Ken Perlin** in 1983. Perlin noise is often used for procedural texture generation in computer graphics.

noise(): A friend of random()

Example 4

w07_code_04

```
int numSlices = 30;
int offset = 10;

void setup() {
  size(600, 400);
  stroke(#ffff00);
  background(0);
  rectMode(CENTER);

  translate(0,100);
  int lastY = int(random(-100,100));
  for (int x = 0; x < width; x += 2) {
    int nowY = int(random(-100,100));
    line(x, lastY, x+2, nowY);
    lastY = nowY;
  }

  translate(0,200);
  lastY = int(map(noise(-0.01), 0,1.0, -100,100));
  for (int x = 0; x < width; x += 2) {
    float n = noise(x * 0.01);
    int nowY = int(map(n, 0,1.0, -100,100));
    line(x, lastY, x+2, nowY);
    lastY = nowY;
  }

}
```



Perlin Noise: noise()

```
noise(<x>);                 // 1D noise
noise(<x>, <y>);            // 2D noise
noise(<x>, <y>, <z>);       // 3D noise
```

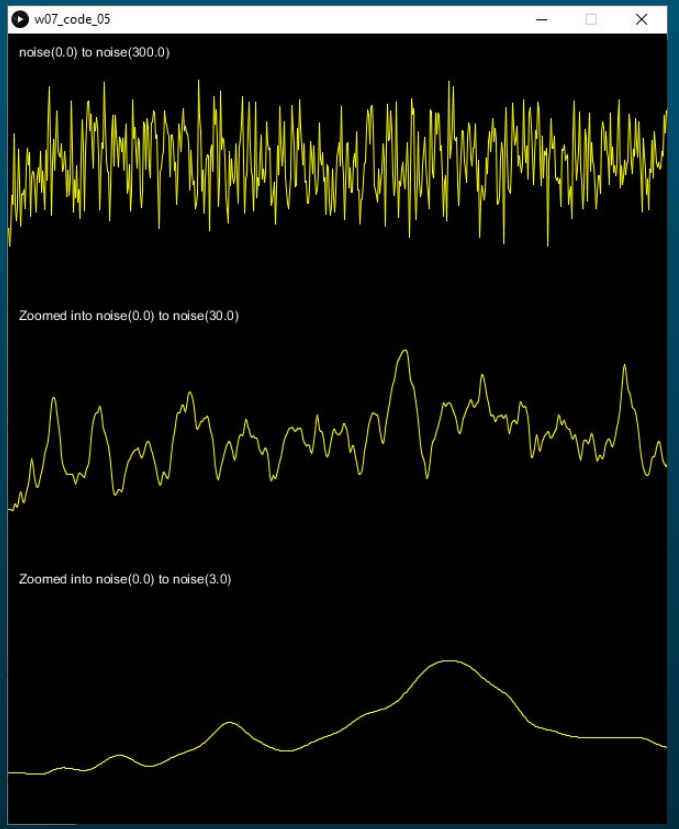
accepts one, two or three numbers as parameter(s). These parameters represent the coordinate of the respective **noise space**. The output of `noise()` always falls into the range of 0.0 to 1.0.

Example 5 - 1D Noise

Example 5

w07_code_05

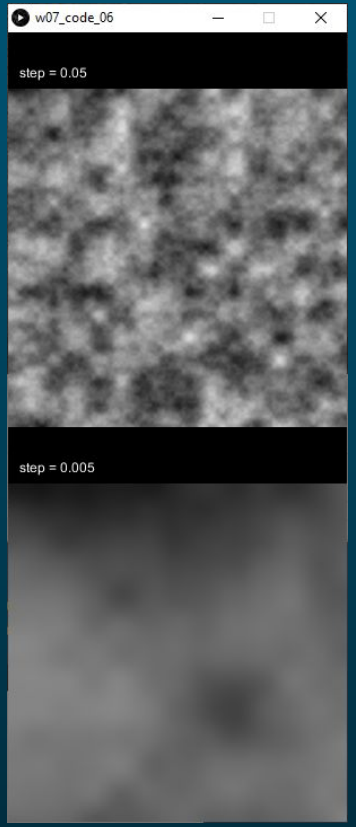
```
void setup() {
  size(600, 720);
  stroke(#ffff00);
  background(0);
  rectMode(CENTER);

  translate(0,120);
  text("noise(0.0) to noise(300.0)", 10, -100);
  plot1DNoise(0.5);

  translate(0,240);
  text("Zoomed into noise(0.0) to noise(30.0)", 10, -100);
  plot1DNoise(0.05);

  translate(0,240);
  text("Zoomed into noise(0.0) to noise(3.0)", 10, -100);
  plot1DNoise(0.005);
}

void plot1DNoise(float step) {
  int lastY = int(map(noise(-step), 0,1.0, -100,100));
  for (int x = 0; x < width; x ++) {
    float n = noise(x * step);
    int nowY = int(map(n, 0,1.0, -100,100));
    line(x, lastY, x+1, nowY);
    lastY = nowY;
  }
}
```



Example 6 - 2D Noise

w07_code_06

```
void setup() {
  size(300, 700);
  stroke(#ffff00);
  background(0);

  text("step = 0.05", 10, 40);
  plot2DNoise(0.05, 50);

  text("step = 0.005", 10, 390);
  plot2DNoise(0.005, 400);

}

void plot2DNoise(float step, int yOffset) {
  for (float y = 0; y < width; y++) {
    for (float x = 0; x < width; x++) {
      int n = int(255.0 * noise(x * step, y * step));
      color c = color(n,n,n);
      set(int(x), int(y + yOffset), c);
    }
  }
}
```



Reproducible Noise: noiseSeed()

noise() relies on random number generation, so it is also possible to create re-producible outputs from noise() functions. noiseSeed(int seed) works similarly as randomSeed(); when you call noiseSeed() with the same parameter before calling noise(), the subsequent outputs from noise() will be produced by using the **same noise space** defined by the seed value.

```
noiseSeed(<int seed>)

// Define the noise space using noiseSeed()
noiseSeed(10);
noise(0.1);
```

Level of Detail: noiseDetail()

noise() function uses multiple octaves of function to create its outputs. In short, the number of octaves define the level of detail of its outputs.

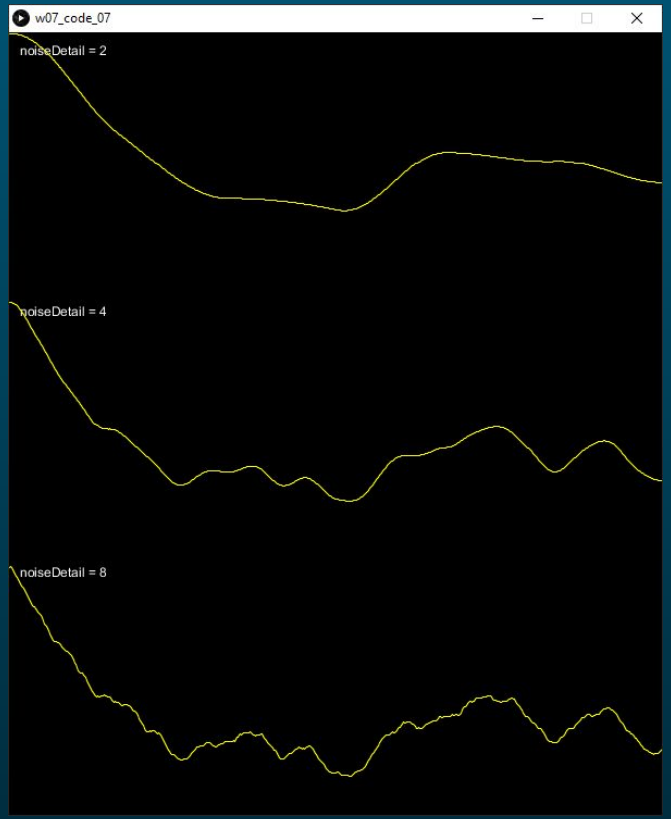
```
noiseDetail(<int lod>)
```

```
// Define the level of details using noiseDetail()
noiseDetail(10);
noise(0.1);
```

Example 7 - Level of Detail

w07_code_07

```
void setup() {
  size(600, 720);
  stroke(#ffff00);
  background(0);
  rectMode(CENTER);

  translate(0,120);
  text("noiseDetail = 2", 10, -100);
  plot1DNoiseLOD(0.005, 2);

  translate(0,240);
  text("noiseDetail = 4", 10, -100);
  plot1DNoiseLOD(0.005, 4);

  translate(0,240);
  text("noiseDetail = 8", 10, -100);
  plot1DNoiseLOD(0.005, 8);
}

void plot1DNoiseLOD(float step, int lod) {
  noiseDetail(lod);
  int lastY = int(map(noise(-step), 0,1.0, -150,150));
  for (int x = 0; x < width; x ++) {
    float n = noise(x * step);
    int nowY = int(map(n, 0,1.0, -150,150));
    line(x, lastY, x+1, nowY);
    lastY = nowY;
  }
}
```



Example 8 - Art of Noise

w07_code_08

```
  size(600, 600);
  PFont myFont = createFont("ChunkFive-Regular.otf", 32);
  textAlign(CENTER, CENTER);
  textFont(myFont);
  textSize(50);
  noiseDetail(5);
}


void draw() {

  background(0);
  float t = frameCount * timeStep;

  for (int n = 1; n <= numWorms; n++) {

    float nx = n * 30.0;
    float ny = n * 60.0;

    for (int v = 1; v < 10; v++) {
      nx = nx + v * 0.004;
      ny = ny + v * 0.004;
      int nowX = int(width * noise(nx + t));
      int nowY = int(height * noise(ny + t));
      fill(v*20, n*50, 0, 220);
      textSize(50 + v*5);
      text(msg[n-1],nowX,nowY);
    }
  }

}
```