

15-104 Introduction to Computing for Creative Practice

Fall 2020

15 Randomness

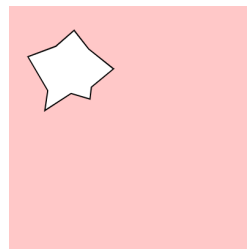
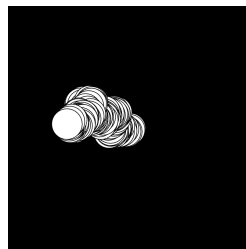
Instructor: Tom Cortina, tcortina@cs.cmu.edu, GHC 4117, 412-268-3514

1



Brownian Motion

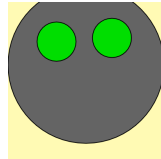
- We have seen several examples of randomness based on calling the `random` function.
- Either we use the number directly, or we use the numbers as offsets to create what is often called a “drunken walk” or **Brownian motion**.



2

Wandering Eyes (Brownian Motion)

```
var leftX = 60;
var rightX = 140;
var leftY = 50;
var rightY = 50;
var noiseParam = 0;
var noiseStep = 0.1;
```



```
function setup() {
  createCanvas(200, 200);
  frameRate(10);
}

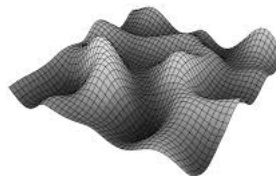
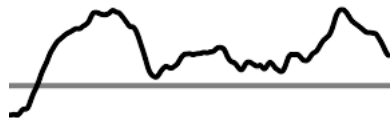
function eye(x, y) {
  fill(0, 222, 0);
  ellipse(x, y, 50, 50);
}
```

```
function draw() {
  background(255, 250, 180);
  fill(100);
  ellipse(100, 80, 200, 200);
  eye(leftX, leftY);
  eye(rightX, rightY);
  leftX += random(-2, 2);
  leftY += random(-2, 2);
  rightX += random(-2, 2);
  rightY += random(-2, 2);
  noiseParam += noiseStep;
};
```


3

Perlin Noise (p5.js reference)

- The `noise` function returns a smoothly varying value as a function of 1, 2, or 3 parameters that can be interpreted as time, space, or both. Perlin noise is a random sequence generator producing a more naturally ordered, harmonic succession of numbers compared to the standard `random()` function.
- The resulting value will always be between 0.0 and 1.0.



4



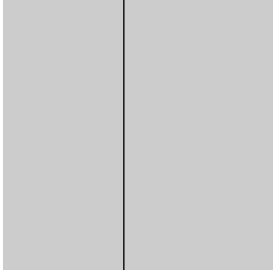
Perlin Noise (p5.js reference)

```


var xoff = 0.0;
function setup() {
  createCanvas(200, 200);
}
function draw() {
  background(204);
  xoff = xoff + 0.01;
  var n = noise(xoff) * width;
  line(n, 0, n, height);
}

```

What do you think happens if you increment the x-offset (xoff) by 0.1 instead of 0.01?



5

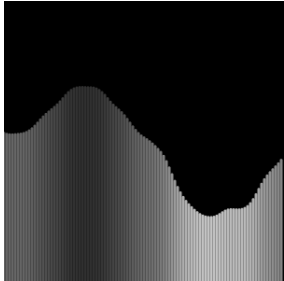


Perlin Noise (p5.js reference)

```


var noiseScale=0.02;
function setup() {
  createCanvas(300, 300);
}
function draw() {
  background(0);
  | for (var x=0; x < width; x++) {
    var noiseVal = noise((mouseX+x)*noiseScale,
                        mouseY*noiseScale);
    stroke(noiseVal*255);
    line(x, mouseY+noiseVal*80, x, height);
  }
}

```



6

Perlin Noise

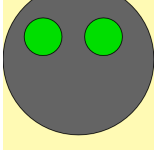


```

var leftX = 60;
var rightX = 140;
var leftY = 50;
var rightY = 50;
var noiseParam = 0;
var noiseStep = 0.1;

function eye(x, y) {
  fill(0, 222, 0);
  var offset = noise(noiseParam);
  offset = map(offset, 0, 1, -20, 20);
  ellipse(x + offset, y, 50, 50);
}

```



```

function setup() {
  createCanvas(200, 200);
  frameRate(10);
}


function draw() {
  background(255, 250, 180);
  fill(100);
  ellipse(100, 80, 200, 200);
  eye(leftX, leftY);
  eye(rightX, rightY);
  noiseParam += noiseStep;
};

```

Perlin noise is a deterministic function: You put in the same parameters and you get out the same value.

7

Perlin Noise

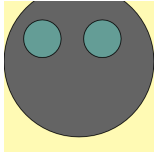


```

var leftX = 60;
var rightX = 140;
var leftY = 50;
var rightY = 50;
var noiseParam = 0;
var noiseStep = 0.1;

function eye(x, y) {
  var offset = noise(noiseParam);
  offset = map(offset, 0, 1, -20, 20);
  fill(0, 222, 0);
  fill(100,
    noise(noiseParam*0.2+100)*255,
    noise(noiseParam*0.2+100)*255);
  ellipse(x + offset, y, 50, 50);
}

```



```

function setup() {
  createCanvas(200, 200);
  frameRate(10);
}

function draw() {
  background(255, 250, 180);
  fill(100);
  ellipse(100, 80, 200, 200);
  eye(leftX, leftY);
  eye(rightX, rightY);
  noiseParam += noiseStep;
};

```

Perlin noise for the eye color.

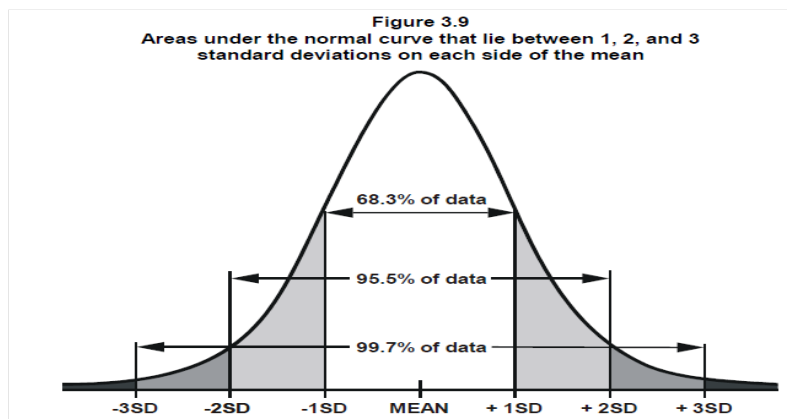
8

Uniform vs. Gaussian Noise

- **Uniform** noise spreads values equally across a fixed range.
 - *Example:* `random(15, 104)` returns a random number in the range [15, 104) with equal probability.*
 - *pseudo-random number generator
- **Gaussian** noise implements a normal distribution, where generated values tend toward a certain average value (called the *mean*), with more or less random variation around the mean determined by the *standard deviation*.
 - *Example:* `randomGaussian()` returns a random number so that over time, the mean is 0 and the standard deviation is 1.
 - *Example:* `randomGaussian(15)` returns a random number so that, over time, the mean is 15 and the standard deviation is 1.
 - *Example:* `randomGaussian(104, 15)` returns a random number so that, over time, the mean is 104 and the standard deviation is 15.


9

Normal Distribution



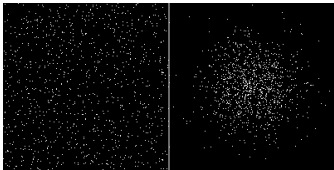
10

Uniform vs. Gaussian Noise



```
function setup() {
  createCanvas(
    400,200);
  background(0);
}


function draw() {
  stroke(255);
  line(width/2, 0, width/2, height); // draw center line
  fill(255); // points are white
  for (var i = 0; i < 1000; i++) {
    // Uniform on the left, Gaussian on the right
    push();
    translate(100,100);
    point(random(-100,100), random(-100, 100));
    pop();
    push();
    translate(300,100);
    point(randomGaussian(0, 30), randomGaussian(0, 30));
    pop();
  }
  noLoop();
}
```



There are probably some Gaussian points that are outside the bounds of the right square. Why?

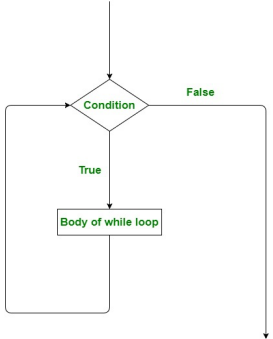
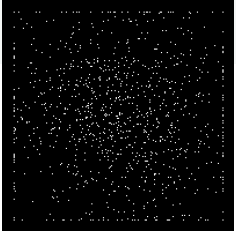
11

Gaussian with Limits



- If the value generated is off the canvas, you could just set it to some default value (e.g. 10 units away from the border), but you will get a distinct border.
- Better solution: keep generating random numbers until you get something in bounds.
- We can use a while loop to do this:


```
while (condition) {
  loop body
}
```

geeksforgeeks.org

12

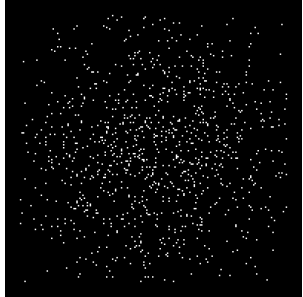
Uniform vs. Gaussian Noise



```

push();
translate(width/2, height/2);
for (var i = 0; i < 1000; i++) {
  // Gaussian noise for both x,y centered on canvas
  var x = randomGaussian(0, 50);
  while (x < -width/2 + 10 || x > width/2 - 10) {
    x = randomGaussian(0, 50);
  }
  // x must be on the canvas now
  var y = randomGaussian(0, 50);
  while (y < -height/2 + 10 || y > height/2 - 10) {
    y = randomGaussian(0, 50);
  }
  // y must be on the canvas now
  point(x, y);
}
pop();


```



The complete draw function is not shown.

13

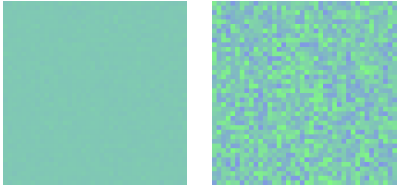
Gaussian for Colors



```

function setup() {
  createCanvas(200,200); frameRate(1);
}
function draw() {
  var range = 50;
  background(0); noStroke();
  for (var x = 0; x < width; x += 5) {
    for (var y = 0; y < height; y += 5) {
      var r = randomGaussian(0, mouseX / 5);
      while (r < -range || r > range) {
        r = randomGaussian(0, 0, mouseX / 5);
      }
      fill(128, 200 + r, 180 - r); // add some random to G & B
      rect(x, y, 5, 5);
    }
  }
}

```



mouseX near 0 mouseX near width

14

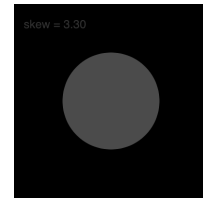
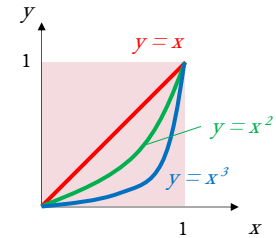
Exponentiation (Power)

You can use exponentiation to scale an effect so that most of the time the effect is small, but once in a while it is large. The higher the power, the more pronounced the effect is. In p5.js, `Math.pow(a, b)` returns a^b .

```
var skew;

function setup() {
  createCanvas(200,200); frameRate(10);
}

function draw() {
  background(0);
  skew = map(mouseX, 0, width, 1, 10);
  fill(Math.pow(random(1), skew) * 255);
  ellipse(width/2, height/2, width/2, height/2);
}
```



How does the program behavior change as you move the mouse left and right?