Chapter 10: Neural Networks

The Nature of Code

Html version

Introduction

Probabilities and random numbers

Uniform

random()

Gaussian/Normal

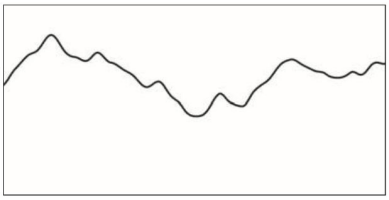
randomGaussian(): -> float

Perlin Noise (smoother approach)

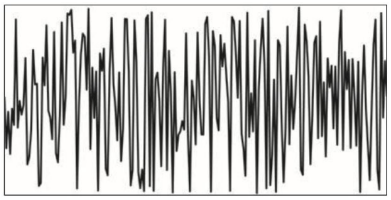
For natural textures

noise()

noiseDetail(): Adjusts the character and level of detail produced by the Perlin noise function



Perlin Noise



Random

Use map() to map random numbers (0, 1) to the range you want

Chapter 1: Vectors

PVector

Reference

Source Code

PVector methods

set()	Set the components of the vector
random2D()	Make a new 2D unit vector with a random direction.
random3D()	Make a new 3D unit vector with a random direction.
fromAngle()	Make a new 2D unit vector from an angle
copy()	Get a copy of the vector
mag()	Calculate the magnitude of the vector
magSq()	Calculate the magnitude of the vector, squared
add()	Adds x, y, and z components to a vector, one vector to another, or two independent vectors
sub()	Subtract x, y, and z components from a vector, one vector from another, or two independent vectors
mult()	Multiply a vector by a scalar
div()	Divide a vector by a scalar
dist()	Calculate the distance between two points
dot()	Calculate the dot product of two vectors
cross()	Calculate and return the cross product
normalize()	Normalize the vector to a length of 1
limit()	Limit the magnitude of the vector
setMag()	Set the magnitude of the vector
heading()	Calculate the angle of rotation for this vector
rotate()	Rotate the vector by an angle (2D only)
lerp()	Linear interpolate the vector to another vector
angleBetween()	Calculate and return the angle between two vectors
array()	Return a representation of the vector as a float array

Vector algebra

Chapter 2: Forces

Static vs. Non-static in Java

Reference

Static method

Static method belong to the class

Non-static method belong to the object

Friction

$$\overrightarrow{Friction} = -1 * \mu * N * \hat{v}$$

Air and Fluid Resistance

General drag equation

$$F_D = \frac{1}{2}\rho v^2 C_D A$$

where

F_D is the **drag force**,
ρ is the density of the fluid,[11]
v is the speed of the object relative to the fluid,
A is the cross sectional area, and
C_D is the drag coefficient – a dimensionless number.

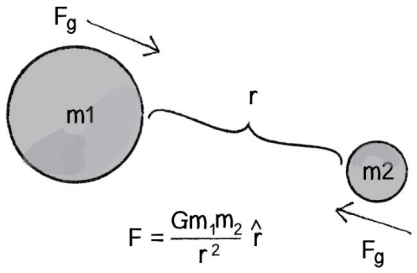
Reduced drag equation

magnitude is speed squared * coefficent of drag

$$F_{drag} = \overbrace{\|v\|^2 * c_d}^{} * \underbrace{\hat{v} * -1}$$

direction is opposite of v (velocity)

Gravitational Attraction



Calculate planetary attraction function

```
PVector attract(Mover m) {
  PVector force = PVector.sub(location,m.location);
  float distance = force.mag();
  distance = constrain(distance,5.0,25.0);

  force.normalize();
  float strength = (G * mass * m.mass) / (distance * distance);
  force.mult(strength);
  return force;
}
```

Remember, we need to constrain the distance so that our circle doesn't spin out of control.

Chapter 3: Oscillation

Angles

radians()/degrees()
rotate(radian)

Angular velocity

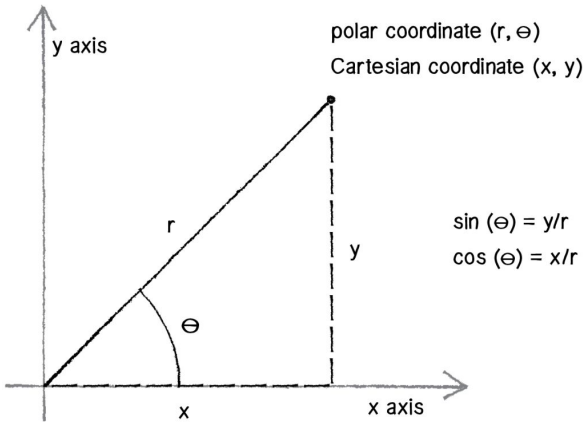
location = location + velocity
velocity = velocity + acceleration

angle = angle + angular velocity

angular velocity = angular velocity + angular acceleration

heading(): Calculates the angle of rotation for a vector (2D vectors only) == atan2()

Polar vs. Cartesian Coordinates



Oscillation

$$y(t) = A\sin(2\pi ft + \varphi) = A\sin(\omega t + \varphi)$$

where:

- A = the *amplitude*, the peak deviation of the function from zero.
- f = the *ordinary frequency*, the *number* of oscillations (cycles) that occur each second of time.
- $\omega = 2\pi f$, the *angular frequency*, the rate of change of the function argument in units of radians per second
- φ = the *phase*, specifies (in radians) where in its cycle the oscillation is at $t = 0$.

Wave

Static continuous wave example

```
float angle = 0;
float angleVel = 0.2;
float amplitude = 100;

size(400,200);
background(255);

stroke(0);
strokeWeight(2);
noFill();

beginShape();
for (int x = 0; x <= width; x += 5) {
  float y = map(sin(angle),-1,1,0,height);

  vertex(x,y);
  angle +=angleVel;
}
endShape();
```

Here's an example of using the map() function instead.

With beginShape() and endShape(), you call vertex() to set all the vertices of your shape.

Sinusoidal standing wave example

```
float startAngle = 0;
float angleVel = 0.1;

void setup() {
  size(400,200);
}

void draw() {
  background(255);
```

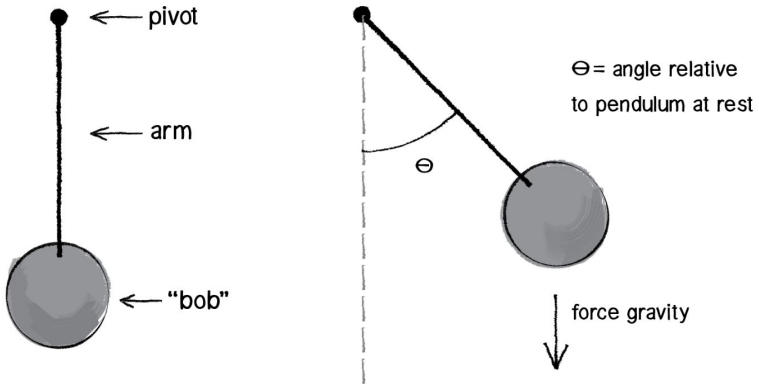
```
float angle = startAngle;
```

In order to move the wave, we start at a different theta value each frame. startAngle += 0.02;

```
  for (int x = 0; x <= width; x += 24) {
    float y = map(sin(angle),-1,1,0,height);
    stroke(0);
    fill(0,50);
    ellipse(x,y,48,48);
    angle += angleVel;
  }
}
```

Simple pendulum

Simple pendulum physics



Spring

Hooke's Law

```
void connect(Bob b) {
```

Calculate spring force—our implementation of Hooke's Law.

```
  PVector force =
    PVector.sub(b.location,anchor);
```

Get a vector pointing from anchor to Bob location.

```
  float d = force.mag();
  float stretch = d - len;
```

Calculate the displacement between distance and rest length.

```
  force.normalize();
  force.mult(-1 * k * stretch);
```

Direction and magnitude together!

```
  b.applyForce(force);
}
```

Call applyForce() right here!

Chapter 4: Particle Systems

ArrayList

Enhanced for loop (no indexing)

```
for (Particle p : particles) {
  p.run();
}
```

This enhanced loop also works for regular arrays!

Remove from list

Wrong example with removing/inserting

```
for (int i = 0; i < particles.size(); i++) {
  Particle p = particles.get(i);
  p.run();
  if (p.isDead()) {
    particles.remove(i);
  }
}
```

If the particle is "dead," we can go ahead and delete it from the list.

Correct example with removing/inserting

```
for (int i = particles.size()-1; i >= 0; i--) {
  Particle p = (Particle) particles.get(i);
  p.run();
  if (p.isDead()) {
    particles.remove(i);
  }
}
```

Looping through the list backwards

Iterator

Declaration

```
Iterator<Particle> it = particles.iterator();
```

Note that with the Iterator object, we can also use the new <ClassName> generics syntax and specify the type that the Iterator will reference.

Iterating (indexing taken care of with removing/inserting)

```
while (it.hasNext()) {
  Particle p = it.next();
  p.run();
}
```

An Iterator object doing the iterating for you

Inheritance and Polymorphism

```
class Dog extends Animal {
   color haircolor;

   Dog() {
     super();
     haircolor = color(random(255));
   }

   void eat() {
     super.eat();

     println("Woof!!!");
   }

   void bark() {
    println("WOOF!");
  }
}
```

Call eat() from Animal. A child can execute a function from the parent while adding its own code.

Add some additional code for a Dog's specific eating characteristics.

Particle system with repellers