Math, Angles and Randomness

Shiffman Chpt 13

Basic Math

- Addition: +
- Subtraction: -
- Division: /
- Multiplication: *
- Assignment: =

float a = 6 + 5 * 4 /2 - 1;

- Follow standard order of operations (result is 15)

Math with Integers

- When working with integers, floating point values will round down
- Examples:
- 3/4 = 0 (integer)
- 2.5 + 2 +. 1 = 4 (integer)
- (This only holds if you are doing integer math. The above would be .75 and 4.6 respectively if you are doing floating point math.)

(Surprising) Results from Integer Math

- Processing will decide if the number is an integer or a float based on whether it has a decimal place
- If neither number has a decimal place, it will perform integer arithmetic. e.g.:

```
float a = 3/4; //integer division, result is 0
float b = 3.0/4;  //floating point division, result is 0.75
println("a is " +a + " b is " + b);
```

  - a is 0.0 b is 0.75

Math Short forms

- Increment: ++
  - a++; is equivalent to a = a+1;
- Decrement: --
  - a--; is equivalent to a = a-1;
- Add to: +=
  - a += 5; is equivalent to a = a+5;
- Multiply by: *=
  - a *= 5; is equivalent to a = a * 5;
- Also: /=, -=

Modulus

- Operator that returns the remainder of division
- A % B = C
  - Divide A by B, and C equals the remainder
  - C is always < B by definition
- Say A mod B
- Examples:

14 % 3 = 2

4 % 7 = 4

Angles

- Trigonometric functions in most programming languages work in *radians* not *degrees*
- 2 π radians = 360 degrees
- radians(); //converts number in degrees to radians
- PI and TWO_PI are defined constants
- If theta is in radians, use sin(theta)
- If theta is in degrees, use sin(radians(theta))

(Pseudo-) Random Numbers

Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. - John von Neumann

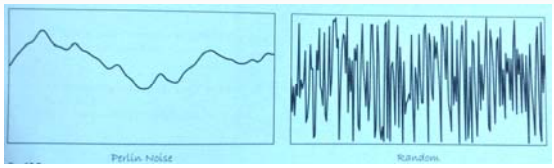
- random(); //returns a pseudorandom number from a uniform distribution

(Pseudo-) Random Numbers

- noise(<time>); //implementation of Perlin noise that returns a number that must be somewhat close to the number for the previous time
  - Number is in [0..1]
  - 1D, 2D or 3D
- noiseDetail(); //control the octaves used in the noise function

noise() vs. random()

- Noise has more gradual change



(From Shiffman)

noise(<time>)

- Always return a value between 0 and 1
- Must increment time to get a different value
- Small increments will cause noise to change slowly
- Larger increments will lead to larger changes

```
float r = random(1);
if(r< prob)
{
        //do something random
}

//rain drops example
```

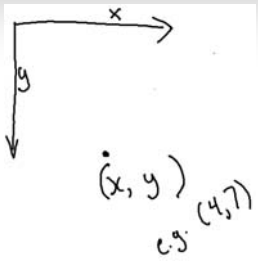
```
ellipse(random(width),random(height),64,64);
```

Seeding the Random Numbers

- noiseSeed(<int>); //used with noise()
- randomSeed(<int>);//used with random()

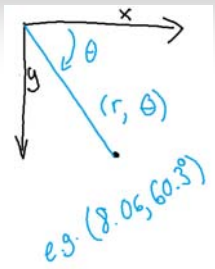
Cartesian and Polar Coordinates

- It takes two numbers to specify a location in 2D
- These could be x-y Cartesian Coordinates



Cartesian and Polar Coordinates

- It takes two numbers to specify a location in 2D
- These could be x-y Cartesian Coordinates
- Or, they could be (r, θ) Polar Coordinates
- r : radius from origin
- θ : angle from axis



Cartesian and Polar Coordinates

- Polar Coordinates are more convenient in certain cases
  - When describing a circle
  - When it is useful to specify an angle from the origin
  - When it is useful to specify a distance from the origin

Cartesian and Polar Coordinates

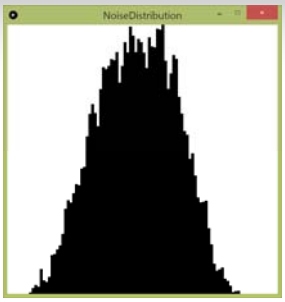
- // Polar to Cartesian conversion

```
float x = r * cos(theta);
float y = r * sin(theta);
```

Path Drawing Example

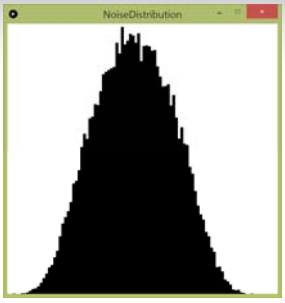
Noise is Biased Around 0.5

- x axis is the value of noise between 0 and 1
  - Divided into 0.01 buckets
- y is the number of times that value occurred in a test run



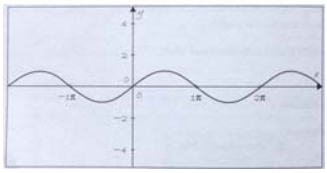
Noise is Biased Around 0.5

Noise is Biased Around 0.5



Trig Functions for Oscillation

- Trig functions provide a nice way to give natural appearing variation
  - e.g. sin(time);



(From Shiffman)

- Pendulum Example