# The Nature of Code

*- Daniel Shiffman -*

*跨院碩一，科技藝術組，陳信甫*

書籍 -

*The Nature of Code*

*Learning Processing*

***Daniel Shiffman***

1973 / 07 /29

計算機程序員

*Processing Foundation*董事會成員

*Daniel Shiffman*目前致力於開發Processing的相關教程跟範例。*Shiffman*也開設了一個受歡迎的YouTube頻道 *The Coding Train*，其中提供了有關如何在Processing跟P5.js中編程的說明性視頻。

## **The Nature of Code** 是一本什麼書？

本書中的內容能教導學生學習程式設計的基礎知識（變數、條件、迴圈、對象、數組），本課程主要遵循我的入門書《*Learning Processing*》中的內容，一旦您瞭解了基礎知識並看到了一系列應用程序，您的下一步可能以是深入研究特定的領域，這本書的目的很簡單。我們想看看在我們所在的世界及自然中發生的事情，然後決定編寫相關程式碼來模擬這類的情況。

那麼這本書到底是什麼？這是一本科學書嗎？答案是一個響亮的否定。沒錯，我們可能會研究來自物理或生物學的課題，但我們的工作不是以特別高的學術嚴謹程度來研究這些課題。取而代之的是，我們瀏覽科學概念，並建構、編程特定軟件示範所需的部分。

# This book as a syllabus

Week 1 Introduction and Vectors (Chapter 1)

Week 2 Forces (Chapter 2)

Week 3 Oscillations (Chapter 3)

Week 4 Particle Systems (Chapter 4)

Week 5 Physics Libraries Part I (Chapter 5)

Week 6 Physics Libraries Part II & Steering (Chapters 5-6)

Week 7 Present midterm projects about motion

Week 8 Complex Systems: Flocking and 1D Cellular Automata (Chapters 6-7)

Week 9 Complex Systems: 2D Cellular Automata and Fractals (Chapters 7-8)

Week 10 Genetic Algorithms (Chapter 9)

Week 11 Neural Networks (Chapter 10)

Weeks 12-13 Final project workshop

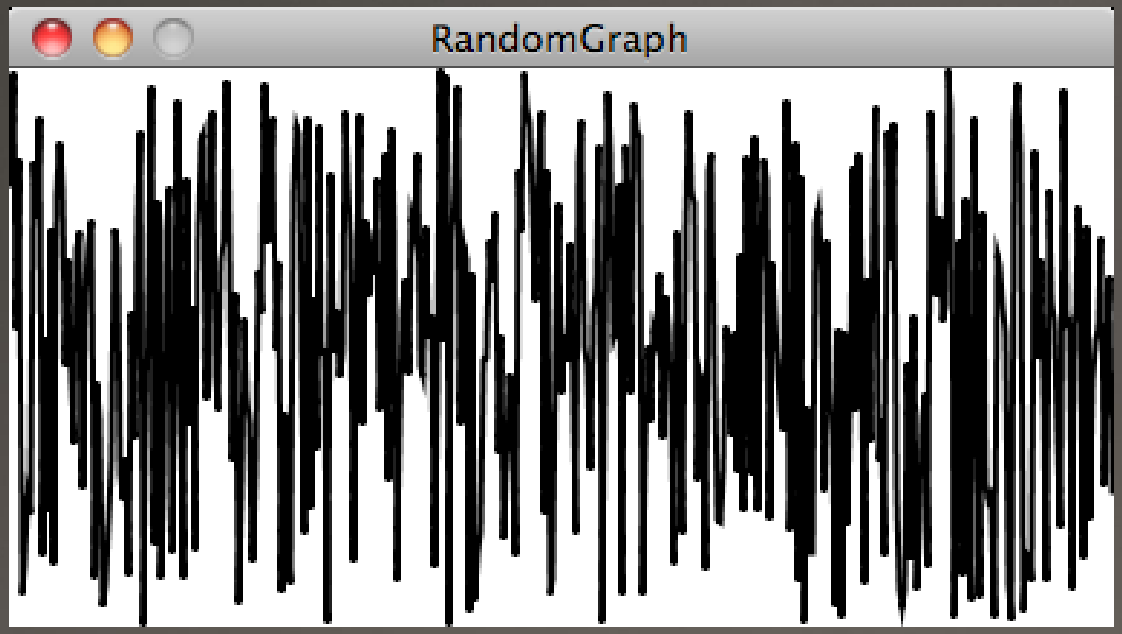
Week 14 Final project presentation

# 進入章節之前的的重要函數 – noise ( ) 和 random ( ) 的區別

noise ( ) ， 是隨機序列發生器，產生比標準 random( ) 函數更自然，它由Ken Perlin在20世紀80年代開發，並已用於圖形應用程序，以生成程序紋理，形狀，地形和其他看似有機的形式。

noise ( ) ，基本上是 random( ) 的一種,但相較於random, noise還帶有方向性,同時noise的數值永遠應介於0到1之間的,所以他就像sin( ),cos( )一般,當我們使用它時,通常都還會給他乘以更大的數來做rescale的效果.

# random ( ) – (無規律的去得到一個隨機值)



關於 random ( ) 的隨機移動方式可以打個比方，想像您正站在平衡木的中間。 每十秒鐘，您擲一枚硬幣。 頭，向前邁出一步。 尾巴，向後退一步。 這就是random的隨機遊走，定義為一系列隨機的路徑。你也可以通過將同一枚硬幣丟兩次來進行二次的隨機遊走

| Flip 1 | Flip 2 | Result |
|---|---|---|
| Heads | Heads | Step forward. |
| Heads | Tails | Step right. |
| Tails | Heads | Step left. |
| Tails | Tails | Step backward. |

1.

讓我們從定義 random walker 開始， random walker 僅需要 兩個變數，一個變數用於 x 位置，一個變數用於 y 位置。

```
class Walker {
  int x;
  int y;
```

Objects have data.

2.

接下來是對這物件的初始設定，你可以將其視為物件的setup（）。在此，我們將初始化Walker的起始位置

```
Walker() {
  x = width/2;
  y = height/2;
}
```

Objects have a constructor where they are initialized.

3 .

接下來是針對給 random walke 的兩項功能 ，首先是呈現在畫面上的黑點跟要出現的位置，接著會隨著後續 random ( ) 的條件式去移動。

```
void display() {
  stroke(0);
  point(x,y);
}
```

Objects have functions.

4 .

接著設定 choice 的變數 ，並賦予random 的數值，選擇0到4之間的隨機浮點數並將其轉換為整數，結果為0、1、2或3。從技術上講，最高的數字永遠不會是 4，而是 3.999999999（與小數位數一樣多的9）。由於轉換成整數的過程會捨棄小數點後的位，因此我們可以獲得的最高整數是3

```
void step() {
  int choice = int(random(4));
```

0、1、2或3

5.

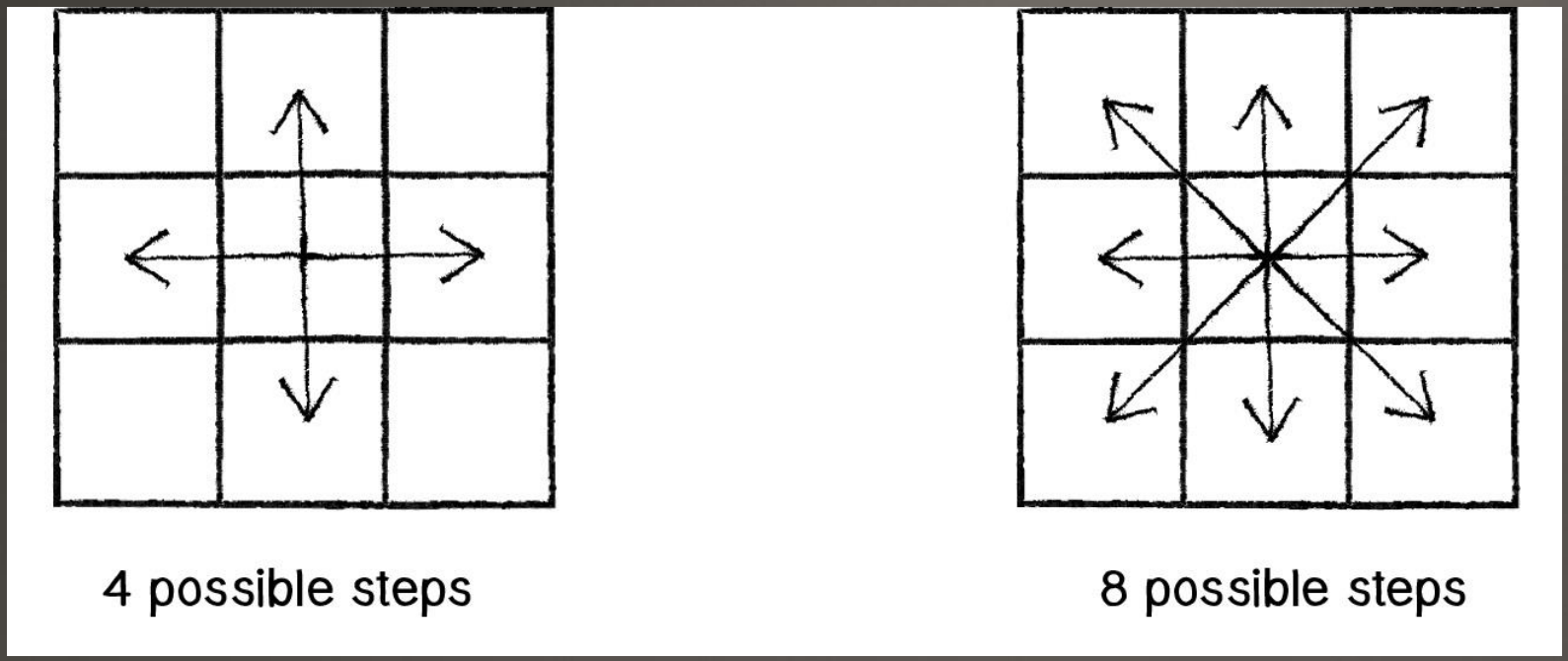
接下來，我們將根據條件式的選擇的數採取適當的步驟（向左，向右，向上或向下）。

```
    if (choice == 0) {                                    隨機的"選擇"決定了我們的步驟。
      x++;
    } else if (choice == 1) {
      x--;
    } else if (choice == 2) {
      y++;
    } else {
      y--;
    }
  }
}
```

6.

接著回去創建視窗並將兩項功能回應。

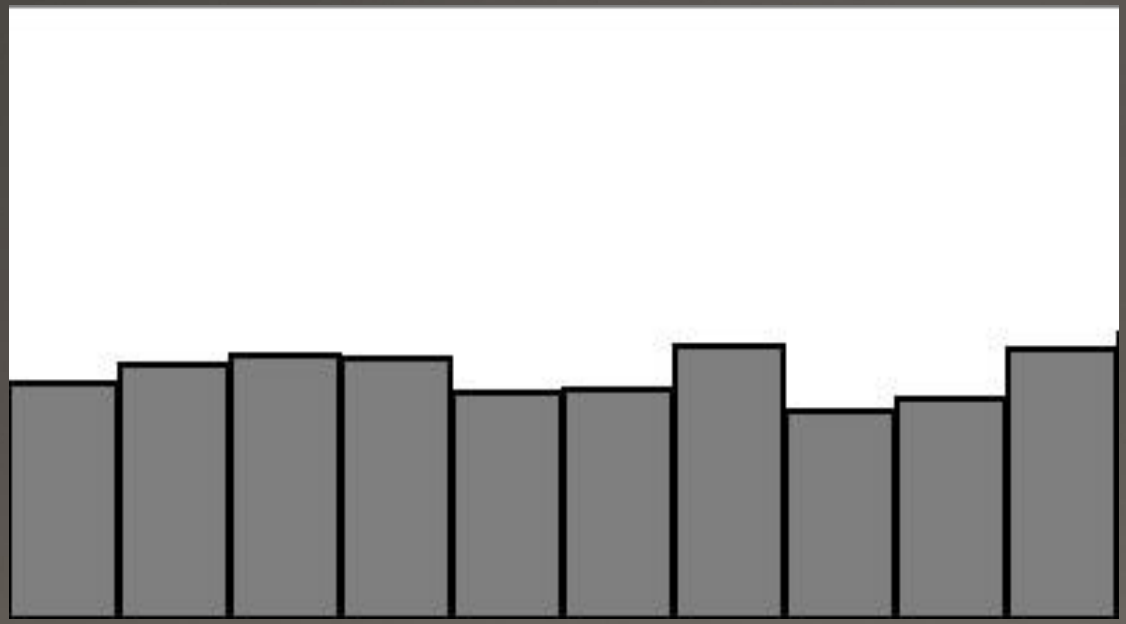
```
void setup() {
  size(640,360);
  w = new Walker();                                       創建沃克。
  background(255);
}

void draw() {
  w.step();                                               Walker上的呼叫功能。
  w.display();
}
```



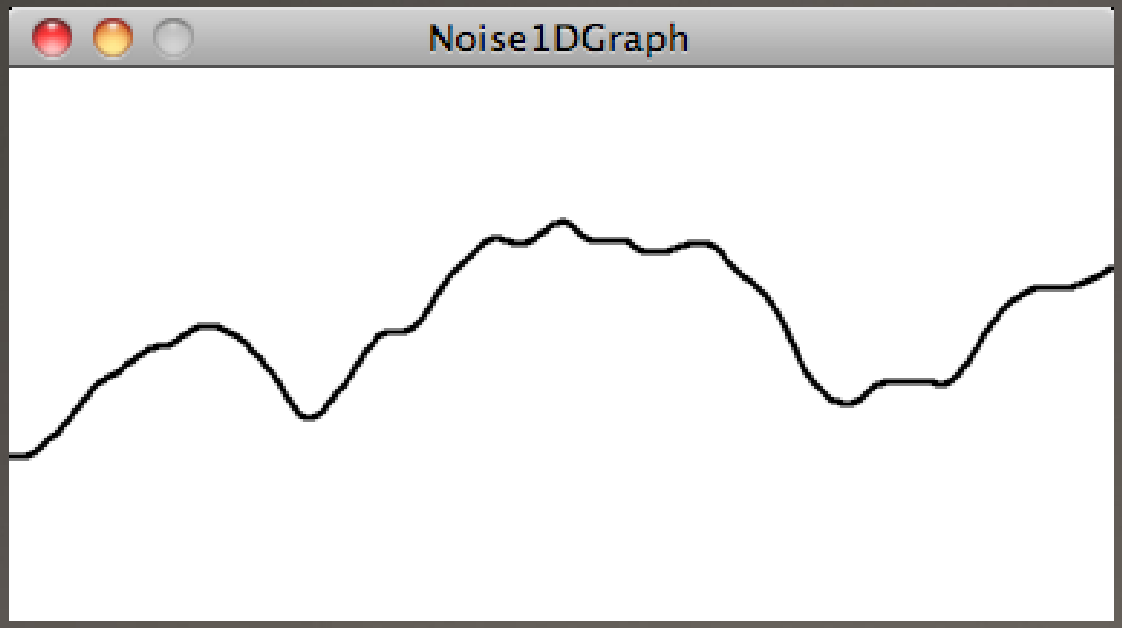

我們可以對隨機功能進行一些改進。 首先，這個 Walker 的方向選擇僅限於四個選項-上，下，左和右。 但是任何像素在視窗中還有八個可能的位置，第九種可能性是留在同一個地點。

random walker 的所有這些變化都有一個共同點：在任何時候，方向上邁出的步伐將在任何方向上邁出的步伐的概率方向。換句話說，如果有四個可能的方向，則有四分之一（25%）的機會採取任何的步驟。或有了九個可能的步驟，這就有9分之一（11.1%）的機會

random（）函數獲得的隨機數並不是真正的隨機數。因此，它們被稱為“偽隨機”。它們是模擬隨機性的數學函數的結果。這個函數會隨時間產生模式，但是這段時間對我們來說是如此之長，就像純隨機性一樣！



# noise ( ) - (無規律的在小範圍裡得到隨機值)



根據 noise ( ) 計算一個介於 0 和 寬度 之間的x值-這不是正確的實現。儘管 random（ ）函數的參數指定了最小值和最大值之間的值範圍，但noise（ ）不能以這種方式工作。取而代之的是，輸出範圍是固定的，它始終返回0到1之間的值。

一旦我們獲得了介於0到1之間的值，就可以將範圍映射(map)到我們想要的值。最簡單的方法是使用map（ ）函數。

```
float t = 0;

void draw() {
  float n = noise(t);
  float x = map(n,0,1,0,width);                    使用map ( ) 自定義Perlin噪聲的範圍
  ellipse(x,180,16,16);

  t += 0.01;
}
```

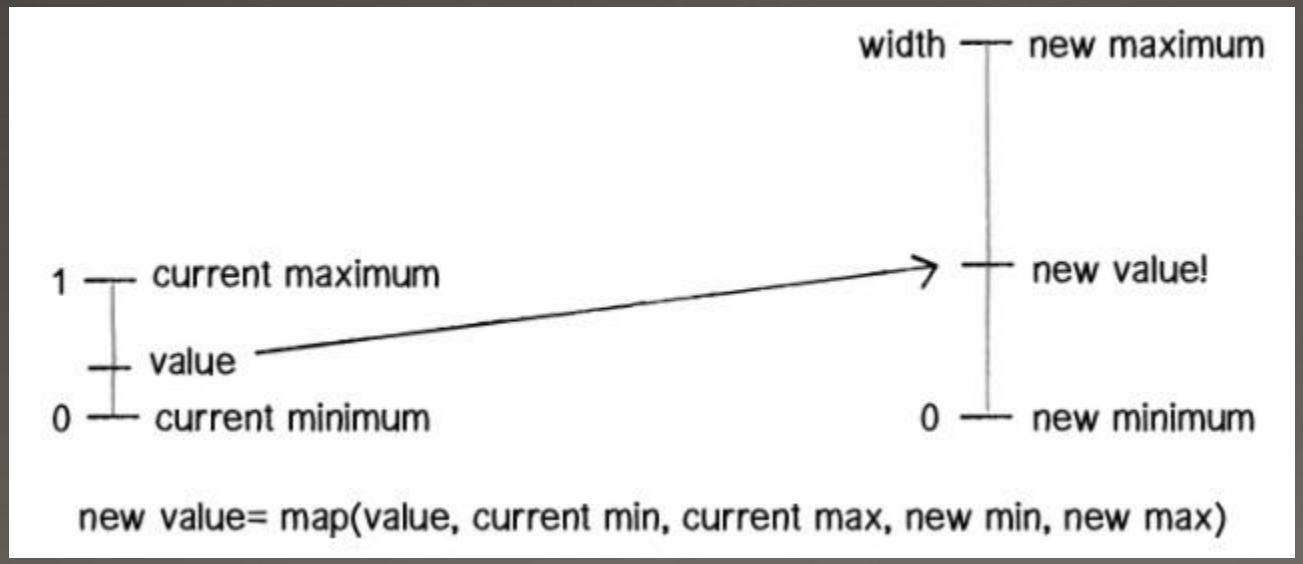
# Noise Walker -

```
class Walker {
  float x,y;

  float tx,ty;

  Walker() {
    tx = 0;
    ty = 10000;
  }

  void step() {
    x = map(noise(tx), 0, 1, 0, width);          x- and y-location mapped from noise
    y = map(noise(ty), 0, 1, 0, height);


    tx += 0.01;                                  Move forward through "time."
    ty += 0.01;
  }
}
```

上面的範例需要另外一對變量：tx和 ty。這是因為我們需要跟踪兩個時間變量，一個用於Walker對象的x位置，另一個用於y位置。但是為什麼tx從0開始， ty從10,000開始？儘管這些數字是任意選擇，但我們非常專門地初始化了兩個具有不同值的時間變量。這是因為 noise 數是確定性的：在特定時間t每次都給您相同的結果。如果我們要求 noise 值在同一時間要兩個 x 和 y，則 x 和 y 始終相等，這意味著Walker對像只會沿對角線移動。所以使用兩個不同部分，x的起始位置為0 ，y的起始位置為10,000，因此x和y似乎表現為彼此獨立。

# 映射 map ( ) ， noise( )

如何處理噪聲值的問題。一旦我們獲得了介於0到1之間的值，就可以將範圍映射到我們想要的值。最簡單的方法是使用Processing的map ( ) 函數。函數有五個參數。首先是我們要映射的值，在這種情況下為n。然後，我們必須為其指定值的當前範圍（最小和最大），然後再加上所需的範圍。



在章節的開頭，我們可能容易陷入使用 noise ( ) 作為函數的功能應用裡。該對象應如何移動？變量要給多少才會有明顯的變化、顏色應用、波形圖形等。

但這並不式要跟說您應該或不應該使用隨機性這項功能 random( )。還是您應該或不應該使用 noise ( ) 。關鍵是程式碼是由您定義，會的功能越多，實現你的圖像作品就會更容易。本書的目標是讓你學會各式各樣的功能。如果您所知道的只有 random( ) ，那麼您的設計思路就會受到限制。當然，noise ( ) 也會對你有所幫助，但您需要更多。