Processing基本教學

1小時快速入門

Processing 官方網站

http://processing.org

下載及安裝Processing

[下載]
http://processing.org/download/

PS.
Windows使用者請下載windows版本，若您知道什麼是JDK並且確定您的電腦已安裝JDK，您可以下載windows [without Java]版本

[安裝]
解壓縮及完成安裝！（綠色軟體）

Sketch

- 我們把每個Processing project視為一個素描(sketch)，而我們所使用的Processing程式語法則為我們的畫筆。因此，我們是用程式在作畫！！
- 每個sketch(也就是每個Processing project)在電腦中是以一個[資料夾]存在，資料夾中存放sketch相關的程式碼(.pde檔)及影音資料(另存放在[data]資料夾中)
- 看範例 (看sketch的資料夾結構)
  - 開啟 File > Examples > Basics >Image > Sprite
  - 查看該sketch的資料夾Sketch > Show Sketch Folder
  - 資料夾內有.pde檔及[data]資料夾 (內有sketch所需的圖片)

Processing是一個連續的畫作

- 用程式作畫最特別的地方是:必須不斷地作畫!!

- 請以電視/電影/卡通來想像
  - 連續播放的影格 → 連續播放的畫作
  - 靜止的畫面:連續播放一模一樣的畫面(影格)
  - 動態的畫面:連續播放不同的畫面(影格)

開始作畫

- 作畫前的準備動作

```
Void setup(){
//...準備動作
}
```

在程式一開始時執行,
只會執行一次

- 開始”不斷地”作畫

```
Void draw(){
//...開始作畫
}
```

緊接在setup()之後執行,
會不斷地一直執行至程式執行結束

何時停止作畫？

- 暫停作畫
  - 在程式中加入delay()敘述
  - 使用noLoop(), 告訴Processing不要不斷地作畫, 只要作畫一次即可
- 完全停止!!
  - 使用者關閉執行/顯示視窗
  - 在程式敘述中, 告訴程式本身自行關閉

```
void setup(){
//...準備動作
}

void draw(){
//...開始不斷地作畫
}

void functionName(){
//...自訂函數
}

void mousePressed(){
//...滑鼠事件控制
}

void keyPressed(){
//...鍵盤事件控制
}

class customeClass {
//...自訂類別
}
```

Processing語法結構

一般型

請copy至PDE執行環境,
並按下Run按鈕來執行看看!!

```
void setup() {
    size(200, 200);
    noStroke();
    background(255);
    fill(0, 102, 153, 204);
    smooth();
    noLoop();
}

void draw() {
    circles(40, 80);
    circles(90, 70);
}

void circles(int x, int y) {
    ellipse(x, y, 50, 50);
    ellipse(x+20, y+20, 60, 60);
}
```

請copy至PDE執行環境,
並按下Run按鈕來執行看看!!

```
void setup() {
    size(200, 200);
    rectMode(CENTER);
    noStroke();
    fill(0, 102, 153, 204);
}

void draw() {
    background(255);
    rect(width-mouseX, height-mouseY, 50, 50);
    rect(mouseX, mouseY, 50, 50);
}
```

Processing語法敘述

函數必須有左右小括號(...)

每行程式敘述必須以分號做結束!!

以左右大括號{...}做為一個程式區塊

```
void setup() {
    size(200, 200);
    rectMode(CENTER);
    noStroke();
    fill(0, 102, 153, 204);
}

void draw() {
    background(255);
    rect(width-mouseX, height-mouseY, 50, 50);
    rect(mouseX, mouseY, 50, 50);
}
```

變數宣告

請copy至PDE執行環境,
然後試著去修改r,g,b,alpha的變數值,
並按下Run按鈕來執行看看!!
看看是否畫面會隨著變數值改變而改變

```
void setup() {
    size(200, 200);
    rectMode(CENTER);
     noStroke();

    int r = 0;
    int g = 102;
    int b = 153;
    int alpha = 204;
    fill(r, g, b, alpha);
}

void draw() {
    background(255);
    rect(width-mouseX, height-mouseY, 50, 50);
    rect(mouseX, mouseY, 50, 50);
}
```

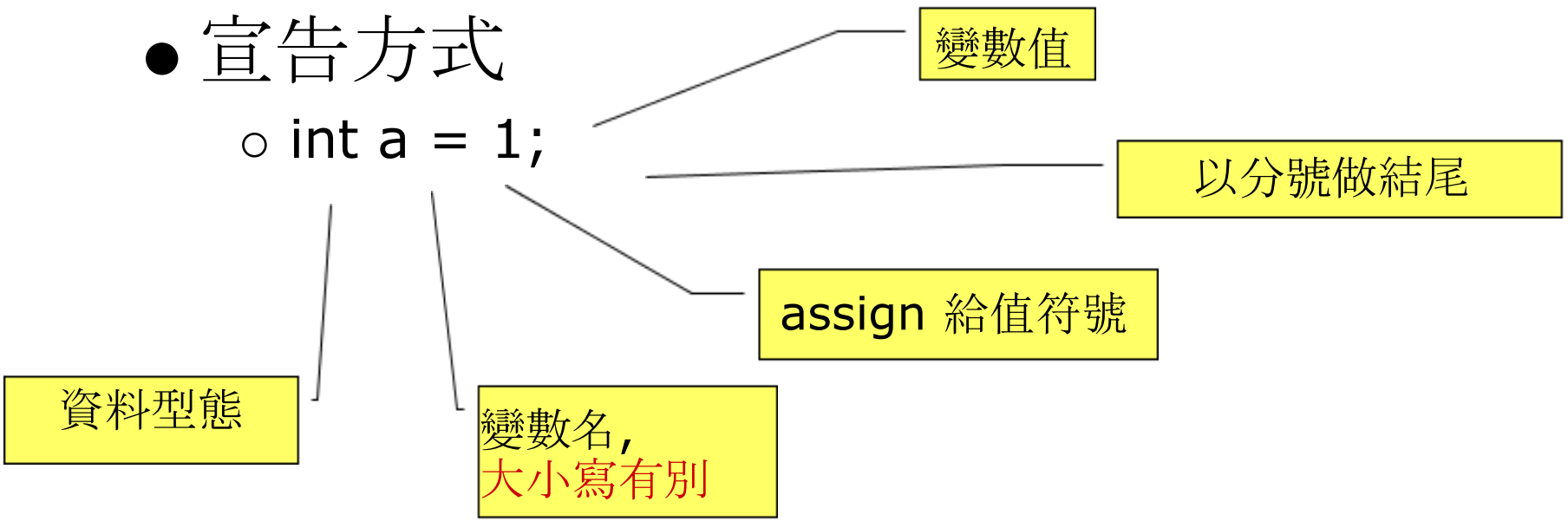
使用整數型態(int)的變數

變數 Variable

- 資料型態(data type)
  - int 整數 例：3, 20, -5, 0
  - float 浮點數 例：1.2301, -0.02
  - String 字串 例：”互動程式設計”
  - boolean 布林值 例：True/False

變數

- 宣告方式
  - int a = 1;



- 範例
  - float discount = 0.85;
  - String myName = ″沈聖博″;
  - boolean hasHomeWork = flase;

變數 Variable

- 在Processing中, 變數的資料型態必須被明確地指出 (這個特性將有助於程式初學者避免不必要的程式錯誤)
- 善用變數, 你的程式就成功一半!!
  - 避免無意義的變數名稱使你的程式難以閱讀 (例如: aaa, bbb)
  - 變數名稱的命名與它本身所代表的變數值相關是最好的! (例如:int xPosition = 5; 這個命名的想法是: 我想宣告一個變數, 用來存放圖片的x座標位置, 這個x座標一開始的值是5, 因此我將變數命名為xPosition, 然後開始進行變數的宣告, 即如範例所述)

變數 Variable

變數命名規則

1. 只可以使用英文字母, 阿拉伯數字, 以及_(底線)
2. 開頭第一個字元, 不能是數字 (錯: 183club, 5566　對: by2)
3. 字母大小寫有別 (SHE, she, She分別代表不同變數)
4. 中間不能有空格 (錯: S B D W, 對: SBDW)
5. 不能使用. (點, dot) (錯: B.A.D)

變數的命名習慣

- 開頭第一個字母為小寫 (例: xPosition, lollipop)
- 駱駝峰式命名 (例: imageRedValue, howTallAreYou)

善用變數, 讓程式更好 閱讀

```
int red = 0;                    //填滿矩形顏色的ARGB值中的R值
int green = 102;                //填滿矩形顏色的ARGB值中的G值
int blue = 153;                 //填滿矩形顏色的ARGB值中的B值
int alpha = 204;                //填滿矩形顏色的ARGB值中的A值, 即透明度
int backgroundGray = 255;    //背景灰階的顏色值

float x1 = 10.0;        //第一個矩形左上角的x座標值
float y1 = 30.5;        //第一個矩形左上角的y座標值
float x2 = 50;          //第二個矩形左上角的x座標值
float y2 = 60.5;        //第二個矩形左上角的y座標值

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

   rectMode(CENTER);
   noStroke();
   fill(red, green, blue, alpha);
}

void draw() {
   background(backgroundGray);
   rect(x1, y1, 50, 50);
   rect(x2, y2, 50, 50);
}
```

//雙斜線後面的一整行是註解, 不會被視為程式碼, 因此不會被執行!

是不是更好閱讀, 更好修改程式碼了呢? 直接從變數名稱就可以知道該變數的變數值是表示什麼意義了!!

變數的使用範圍(生命週期)

變數從哪裡出生(被宣告), 便屬於那裡的區域, 以及那個區域往下所轄的範圍

首先, 我們必須了解何謂範圍:

- 整個Processing程式本身為最大範圍 (全域 global scope)
- 被大括號{...}括起來的區域為一個範圍 (區域 local scope)
  - 函數呼叫
  - for迴圈/while迴圈的使用
  - if/else或switch的判斷
- 範圍可以是巢狀包覆的

變數在各個範圍中的使用順序:
由最小的範圍, 向它的上一層範圍尋找, 找到的第一個相同名稱的變數 (由小範圍找到大範圍)

範例:File → Examples → Basics → Data → VaraibleScope

全域範圍

```
int a = 20;  //宣告一個全域變數a
```

第一層區域範圍

```
void setup()
{
  size(200, 200);
  background(51);
  stroke(255);
  noLoop();
}
```

```
void draw()
{
  line(a, 0, a, height);     //使用全域變數a來畫直線
```

第二層區域範圍

```
  for(int a=50; a<80; a += 2) { //在for迴圈之內, 新宣告一個區域變數a
    line(a, 0, a, height);  //利用for迴圈裡面的區域變數a來畫直線
  }

  int a = 100;  //為draw()函數新宣告一個區域變數a
  line(a, 0, a, height);  //利用draw()裡面的區域變數a來畫直線

  drawAnotherLine();     //呼叫自訂函數drawAnotherLine()

  drawYetAnotherLine(); //呼叫自訂函數setYetAnotherLine()
}
```

```
void drawAnotherLine()
{
  int a = 185;  //為drawAnotherLine()新宣告一個區域變數a
  line(a, 0, a, height);     //利用drawAnotherLine()裡面的區域變數a來畫直線
}
```

```
void drawYetAnotherLine()
{
  line(a+2, 0, a+2, height);     //利用全域變數a來畫直線
}
```

休息一下，我們來點實際的應用

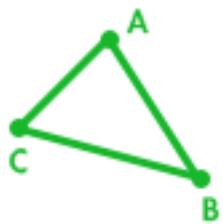
→ 畫幾何圖形

了解從哪裡開始下筆作畫 -- 座標

<u>座標</u>
在畫面上的每一個像素(點)都有其座標位置, 利用座標的表示法, 讓你知道要從何處下筆作畫

例如:
畫直線 (任兩點(A,B)決定一條直線)

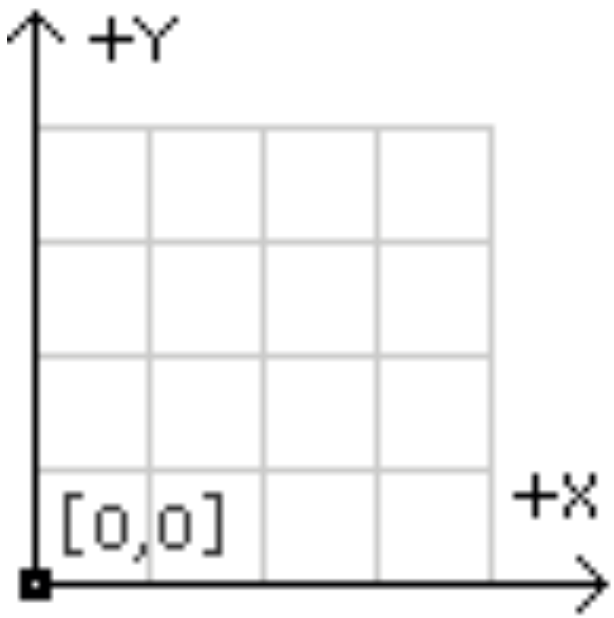
畫三角形 (任意不共線的三點(A,B,C)決定一個三角形)

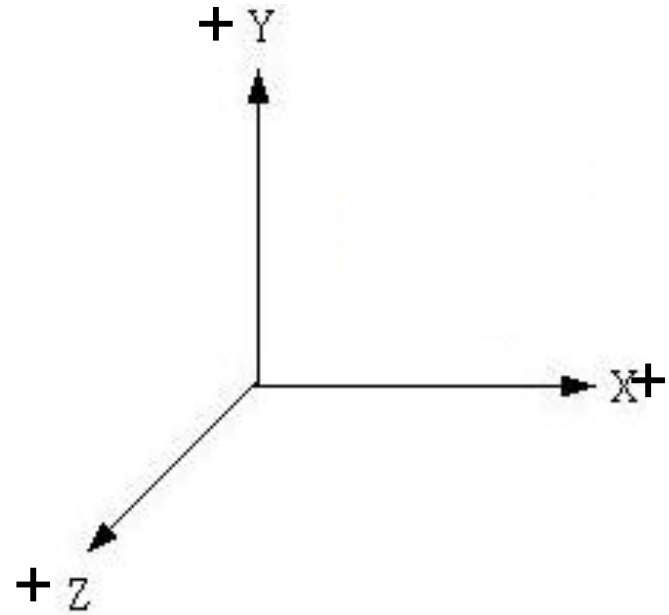


畫圓形 (一個點(A)及一個半徑(r)可劃一個圓)

數學座標

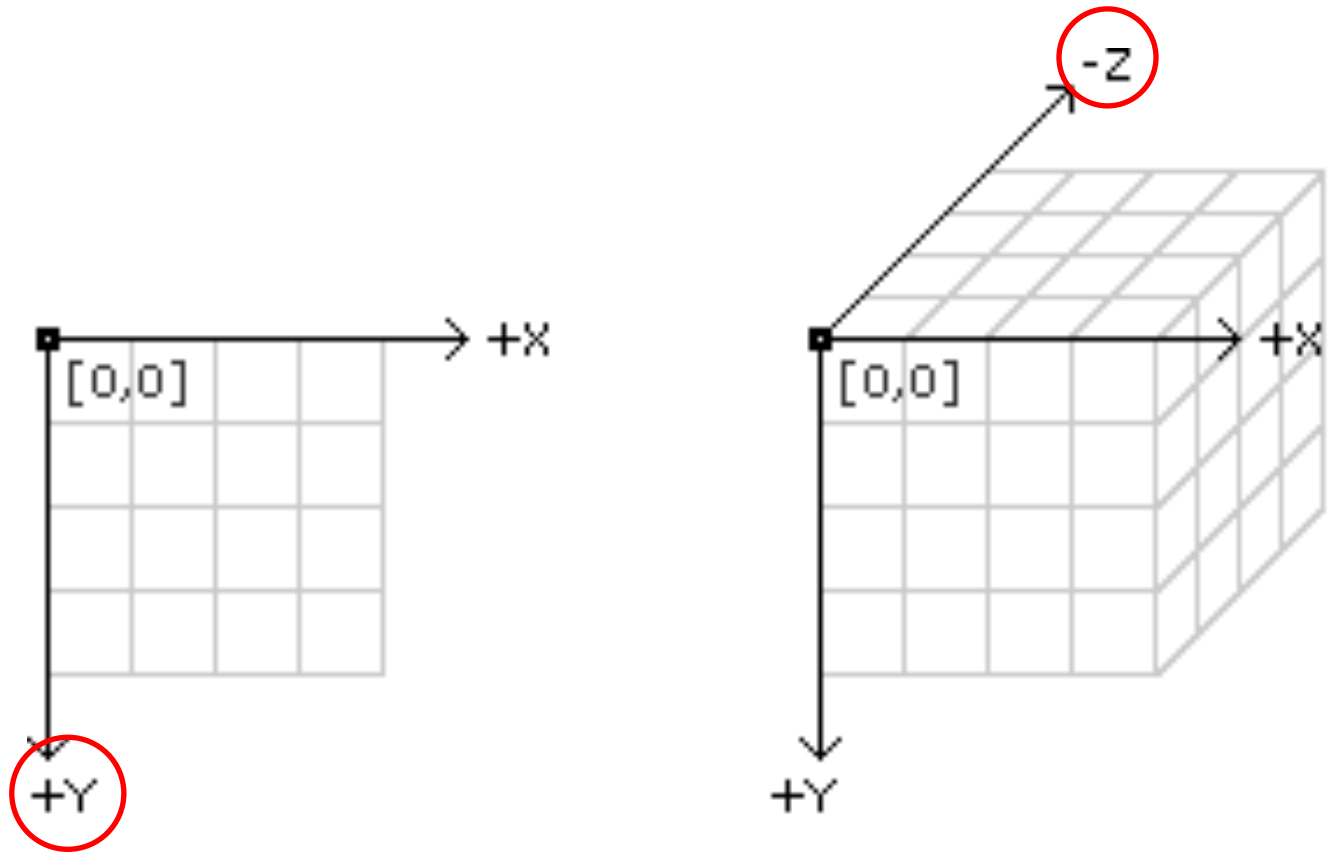
在數學上, X軸是向右漸增, Y軸是向上漸增, Z軸是向外漸增





程式世界裡的座標系統

在程式裡, X軸是向右漸增, Y軸是向下漸增, Z軸是向外漸增



簡單幾何圖形

在Processing中畫基本的幾何圖形, 非常地簡單,
只要下指令(函數呼叫)就可以了, 指令請參考
http://www.processing.org/reference/

Shape

2D Primitives

arc()	曲線
ellipse()	橢圓形
line()	直線
point()	點
quad()	四邊形
rect()	矩形
triangle()	三角形

簡單幾何圖形範例

請copy至PDE執行環境,
並按下Run按鈕來執行看看!!

```
void setup(){
  size(400, 400);
  background(100);
}

void draw(){
  stroke(255);                    //設定邊線顏色為白色
  line(10, 10, 300, 350);         //畫直線

  noFill();                       //設定不填滿
  stroke(0, 255, 0);              //設定邊線顏色為綠色
  ellipse(300, 300, 100, 30);     //畫橢圓形

  fill(random(256), random(256), random(256));    //設定填滿的顏色為隨機的
  noStroke();         //設定不使用邊線
  ellipse(80, 200, 50, 50);     //畫圓形

  noFill();                   //設定不填滿
  stroke(0, 0, 255);      //設定邊線為藍色
  triangle(20, 10, 120, 80, 80, 350);     //畫三角形

  fill(255, 0, 0, 128);     //設定填滿顏色為紅色, 但透明度只有128 (即50%)
  stroke(0, 255, 0);       //設定邊線顏色為綠色
  rect(100, 100, 150, 30);     //畫矩形
}
```

人生會遇到抉擇，程式也會！

如果 … 我就 … 不然的話 …

邏輯式的思考方式

邏輯判斷的結果只有兩種：True 或 False

1. 邏輯判斷來自於比較

意義	Processing語法
小於	<
等於	==
大於	>
不等於	!=
大於或等於	>=
小於或等於	<=

2. 邏輯判斷來自於邏輯運算

意義	Processing語法
AND	&&
OR	||
NOT	!

邏輯式的思考方式

真值表（Truth Table）
(T代表True, F代表False)

A	B	AND
T	T	T
T	F	F
F	T	F
F	F	F

A	B	OR
T	T	T
T	F	T
F	T	T
F	F	F

A	NOT
T	F
F	T

做決定的思考方式

根據邏輯判斷來作決定, 換句話說便是:
滿足某個條件後才作相對應的事

語法一:

```
if(條件判斷){
    //滿足條件的話, 則執行這個區域的程式碼
}
```

語法二:

```
if(條件判斷){
    //滿足條件的話, 則執行這個區域的程式碼
}else{
    //不滿足條件的話, 則執行這個區域的程式碼
}
```

巢狀語法:

```
if(條件判斷1){
    //滿足條件1的話, 則執行這個區域的程式碼
}else if(條件判斷2){
    //滿足條件2的話, 則執行這個區域的程式碼
}else{
    //都不滿足的話, 則執行這個區域的程式碼
}
```

範例

```
void setup(){
  size(400, 400);    //設定畫布大小為400x400
}

void draw(){
  if(mouseX < 100){    //mouseX用來表示目前滑鼠的x座標值
    background(255);  //設定背景為白色
  }else if(mouseX < 300){
    background(128);      //設定背景為灰色
  }else{
    background(0);      //設定背景為黑色
  }

  stroke(255, 0, 0);              //設定畫筆為紅色
  line(100, 0, 100, 400);    //畫直線
  line(300, 0, 300, 400);    //畫直線
}
```

請copy至PDE執行環境,
並按下Run按鈕來執行看看!!

有時候會想再來一次,
程式也會想, 而且可以再來很多次！

重覆做事的方法

有時候我們會希望程式能夠聰明一點，只要我們告訴它從什麼時候開始，以及什麼時候結束之後，它便會乖乖地去做這段時間之內應該做的事，因此，我們就不必一直重覆吩咐同樣一件事情

主要語法有：
1. for-loop
2. while-loop

(ps. 以上語法均可以使用巢狀結構，也可同時使用)

for-loop語法

分號不要忘記！！

```
for (設定判斷的初始值; 邏輯/條件判斷; 更新判斷的數值) {
  //滿足條件(邏輯判斷結果為true)的話, 則執行這個區域的程式碼
}
```

執行過程:

1. 設定判斷的初始值
2. 進行邏輯判斷
3. 如果邏輯判斷的結果為true, 則跳至步驟4; 若結果為false, 則跳至步驟6
4. 執行for{...} 區域的程式碼
5. 更新判斷的數值, 然後跳至步驟2
6. 結束for-loop

for-loop範例（一般型）

```
void setup(){
  size(200, 200);
  background(0);
}

void draw(){
  for(int i=10; i<width; i=i+10) {
    if(i%20 == 0) {       //百分比符號代表運算後取"餘數"
      stroke(153);
      line(i, 40, i, height/2);
    } else {
      stroke(102);
      line(i, 20, i, 180);
    }
  }
}
```

while-loop語法

```
while (邏輯/條件判斷) {
  /*
  滿足條件(邏輯判斷結果為true)的話, 則執行這個區域的程式碼,
  直到條件不滿足(邏輯判斷結果為false)為止
  */
}
```

for-loop改寫成while-loop即為 :

```
設定判斷的初始值
while (邏輯/條件判斷) {
  //do something...
  更新判斷的數值
}
```

while-loop範例

```
void setup(){
  size(200, 200);
  background(255);
}


void draw(){
  int i=0;
  while(i < 80) {
    line(30, i, 80, i);
    i = i + 5;
  }
}
```

範例

請copy至PDE執行環境,
並按下Run按鈕來執行看看!!

```
int k;

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

void draw(){
  background(102);

  // Draw gray bars
  fill(255);
  k=60;
  for(int i=0; i < mouseX/20; i++) {    //重複次數由滑鼠的x座標位置(除以20)決定
    rect(25, k, 155, 5);
    k= k + 10;
  }

  // Black bars
  fill(51);
  k = 180;
  for(int i=0; i < mouseY/15; i++) {    //重複次數由滑鼠的y座標位置(除以15)決定
    rect(105, k, 30, 5);
    k = k - 10;
  }
}
```

OK！人生還很長，你可以學更多！

更多參考資料

[Processing官網]
http://processing.org/exhibition/
http://processing.org/reference/
http://processing.org/learning/
http://processing.org/hacks/

[Processing作品收集]
http://www.processingblogs.org/
http://www.openprocessing.org/

[Processing範例搜尋]
http://builtwithprocessing.org/browser/browse.php

[網頁動畫/JavaScript]
http://processingjs.org/