Programmare in Processing*

Davide Rocchesso

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License †

Abstract

Lezione introduttiva sulla utilizzazione del linguaggio e ambiente Processing per l'insegnamento dell'elaborazione di media e dell'interaction design.

1 Introduzione

NOTE: Questa introduzione è basata sul tutorial di Daniel Shiffman [1] .

Processing è un linguaggio ed un ambiente di sviluppo orientato all' **interaction design** . Nel corso **Elaborazione di Media in Processing**, Processing è uno degli strumenti principali utilizzati per introdurre elementi di elaborazione di suoni e immagini. Processing è una estensione di Java che supporta molte delle strutture Java con una sintassi semplificata.

Processing può essere utilizzato in tre:

Modi di Programmazione

Basic - Sequenza di comandi per il disegno di primitive grafiche –

applet senza naso[2]	`size(256,256);`
	`background(0);`
	`stroke(255);`
	`ellipseMode(CORNER);`
	`ellipse(72,100,110,130);`
	`triangle(88,100,168,100,128,50);`
	`stroke(140);`
	`strokeWeight(4);`
	`line(96,150,112,150);`
	`line(150,150,166,150);`
	`line(120,200,136,200);`

Table 1

*Version 1.16: May 15, 2008 6:40 am -0500
†http://creativecommons.org/licenses/by/2.0/
[1]http://www.shiffman.net/itp/classes/ppaint/

Intermediate - Programmazione procedurale –

[2] http://cnx.org/content/m12614/latest/pinocchiononose.html

applet con naso[3]	`void setup() {`
	`  size(256,256);`
	`  background(0);`
	`}`
	
	`void draw() {`
	`  stroke(255);`
	`  strokeWeight(1);`
	`  ellipseMode(CORNER);`
	`  ellipse(72,100,110,130);`
	`  triangle(88,100,168,100,128,50);`
	`  stroke(140);`
	`  beginShape(TRIANGLES);`
	`  vertex(114, 180);`
	`  vertex(mouseX, mouseY);`
	`  vertex(140, 180);`
	`  endShape();`
	`  strokeWeight(4);`
	`  line(96,150,112,150);`
	`  line(150,150,166,150);`
	`  line(120,200,136,200);`
	`}`

Table 2

Complex - Programmazione Orientata agli Oggetti (Java) –

[3] http://cnx.org/content/m12614/latest/pinocchionose.html

applet con naso colorato[4]	`Puppet pinocchio;`
	
	`void setup() {`
	`  size(256,256);`
	`  background(0);`
	`  color tempcolor = color(255,0,0);`
	`  pinocchio = new Puppet(tempcolor);`
	`}`
	`void draw() {`
	`  background(0);`
	`  pinocchio.draw();`
	`}`
	`class Puppet  {`
	`  color colore;`
	`  Puppet(color c_) {`
	`  colore = c_;`
	`  }`
	`  void draw () {`
	`  stroke(255);`
	`  strokeWeight(1);`
	`  ellipseMode(CORNER);`
	`  ellipse(72,100,110,130);`
	`  stroke(colore);`
	`  beginShape(TRIANGLES);`
	`  vertex(114, 180);`
	`  vertex(mouseX, mouseY);`
	`  vertex(140, 180);`
	`  endShape();`
	`  strokeWeight(4);`
	`  line(96,150,112,150);`
	`  line(150,150,166,150);`
	`  }`
	`}`

Table 3

I programmi Processing possono essere convertiti in applet Java. Per fare ciò è sufficiente andare nel menu **File** e scegliere **Export** . Il risultato finale sarà dunque la creazione di cinque file, inseriti nel folder `applet`:

- **index.html** - sorgente html per visualizzare la applet
- **filename.jar** - la applet compilata, completa di tutti i dati (immagini, suoni, ecc.) necessari
- **filename.pde** - il codice sorgente Processing
- **filename.java** - il codice Java che incorpora il codice sorgente Processing
- **loading.gif** - un'immagine che viene mostrata mentre si attende il caricamento della applet.

Inoltre, mediante **Export Application** è possibile generare una applicazione eseguibile per piattaforma Linux, MacOS, o Windows.

[4] http://cnx.org/content/m12614/latest/pinocchioclassy.html

2 Tipi di Dati

2.1 Variabili

Una variabile è un puntatore a una locazione di memoria, e può riferirsi a valori primitivi (`int`, `float`, ecc.) oppure ad oggetti o array (tabelle di elementi di un tipo primitivo).

L'operazione di assegnamento `b = a` produce

- Se le variabili si riferiscono a tipi primitivi, la copia del contenuto di `a` in `b`.
- Se le variabili si riferiscono a oggetti o array, la creazione di un nuovo riferimento allo stesso oggetto o array.

NOTE: Per avere chiaro il significato di termini informatici quali, ad esempio, quelli che seguono, si consiglia di cercare in Wikipedia[5]

Definition 1: scope
all'interno di un programma, regione in cui si può accedere ad una variabile e modificarne il valore

Definition 2: global scope
definita fuori da setup() e draw(), la variabile è visibile e usabile ovunque nel programma

Definition 3: local scope
definita all'interno di un blocco di codice o di una funzione, la variabile assume valori locali al blocco o alla funzione, ed eventuali valori assunti da una variabile globale omonima vengono ignorati

Example 1: Dichiarazione e allocazione di un array

```
int[] arrayDiInteri = new int[10];
```

3 Strutture di Programmazione

3.1 Istruzioni Condizionali

- if:

```
if (i == NMAX) {
   println("finito");
   }
else {
   i++;
   }
```

3.2 Iterazioni

- while:

```
int i = 0; //contatore intero
while (i < 10) { //scrivi i numeri da 0 a 9
  println("i = "+ i);
```

[5] http://wikipedia.org/

```
    i++;
  }
```

- for:

```
  for (int i = 0; i < 10; i++) { //scrivi i numeri da 0 a 9
    println("i = "+ i);
  }
```

Example 2: Inizializzazione di una tabella di numeri casuali

```
int MAX = 10;
float[] tabella = new float[MAX];
for (int i = 0; i < MAX; i++)
   tabella[i] = random(1); //random numbers between 0 and 1
println(tabella.length + " elementi:");
        println(tabella);
```

4 Funzioni

Le funzioni consentono un approccio modulare alla programmazione. In Processing, in modalità di programmazione **intermediate**, si possono definire funzioni oltre alla `setup()` e `draw()`, usabili all'interno della `setup()` e della `draw()`.

Example 3: Esempio di Funzione

```
int raddoppia(int i) {
   return 2*i;
}
```

Una funzione è caratterizzata dalle seguenti entità (con riferimento all'esempio (Example 3: Esempio di Funzione)) :

- tipo di ritorno (`int`)
- nome di funzione (`raddoppia`)
- parametri (`i`)
- corpo della funzione (`return 2*i`)

5 Classi e Oggetti

Una classe è definita da un insieme di dati e funzioni. Un oggetto è una istanza di una classe. Viceversa, una classe è una descrizione astratta di un insieme di oggetti.

NOTE: Per una introduzione ai concetti di oggetto e di classe si veda Objects and Classes[6].

Example 4: Esempio di Classe

```
Dot myDot;
void setup() {
  size(300,20);
  colorMode(RGB,255,255,255,100);
  color tempcolor = color(255,0,0);
  myDot = new Dot(tempcolor,0);
}

void draw() {
  background(0);
  myDot.draw(10);
}

class Dot
{

  color colore;
  int posizione;

  //****CONSTRUCTOR*****//
  Dot(color c_, int xp) {
    colore = c_;
    posizione = xp;
  }

  void draw (int ypos)    {
    rectMode(CENTER);
    fill(colore);
    rect(posizione,ypos,20,10);
  }
}
```

Una classe è caratterizzata dalle seguenti entità (con riferimento all'esempio (Example 4: Esempio di Classe)) :

- nome di classe (`Dot`)
- dati (`colore, posizione`)
- costruttore (`Dot()`)
- funzioni (o metodi, `draw()`)

[6] "Objects and Classes" <http://cnx.org/content/m11708/latest/>

Un oggetto (istanza di una classe) è dichiarato nello stesso modo in cui si dichiara una variabile, ma ad esso va poi allocato uno spazio (come si è visto per gli array) tramite il suo costruttore (con riferimento all'esempio (Example 4: Esempio di Classe)).

- Dichiarazione: (`Dot myDot;`)
- Allocazione: (`myDot = new Dot(tempcolor,0)`)
- Utilizzazione: (`myDot.draw(10);`)

NOTE: Per una introduzione alla sintassi Java si veda Java Syntax Primer[7]

Exercise 1 ***(Solution on p. 12.)***

Con il metodo `draw()` seguente si vuole dipingere lo sfondo della finestra di un grigio la cui intensità dipenda dalla posizione orizzontale del puntatore del mouse.

```
void draw() {
    background((mouseX/100)*255);
  }
```

Il codice però non fa quello che ci si aspetta. Perché?

Exercise 2 ***(Solution on p. 12.)***

Cosa stampa il seguente frammento di codice?

```
int[] a = new int[10];
a[7] = 7;
int[] b = a;
println(b[7]);
b[7] = 8;
println(a[7]);
int c = 7;
int d = c;
println(d);
d = 8;
println(c);
```

Exercise 3 ***(Solution on p. 12.)***

Il seguente sketch si pone l'obbiettivo di generare un set di 100 cerchi in movimento e di disegnare nella finestra di Processing tutte le corde congiungenti i due punti di intersezione di tutte le coppie di cerchi a intersezione non nulla.

```
/*

   Structure 3

   A surface filled with one hundred medium to small sized circles.
   Each circle has a different size and direction, but moves at the same slow rate.
   Display:
   A. The instantaneous intersections of the circles
```

[7] "Java Syntax Primer" <http://cnx.org/content/m11791/latest/>

```
   B. The aggregate intersections of the circles

   Implemented by Casey Reas <http://groupc.net>
   8 March 2004
   Processing v.68 <http://processing.org>

   modified by Pietro Polotti
   28 March, 2006
   Processing v.107 <http://processing.org>


*/

int numCircle = 100;
Circle[] circles = new Circle[numCircle];

void setup()
{
  size(800, 600);
  frameRate(50);
  for(int i=0; i<numCircle; i++) {
    circles[i] = new Circle(random(width),
      (float)height/(float)numCircle * i,
      int(random(2, 6))*10, random(-0.25, 0.25),
      random(-0.25, 0.25), i);
  }
  ellipseMode(CENTER_RADIUS);
  background(255);
}


void draw()
{
  background(255);
  stroke(0);


  for(int i=0; i<numCircle; i++) {
    circles[i].update();
  }
  for(int i=0; i<numCircle; i++) {
    circles[i].move();
  }
  for(int i=0; i<numCircle; i++) {
    circles[i].makepoint();
  }
  noFill();
}


class Circle
```

```
{
  float x, y, r, r2, sp, ysp;
  int id;

  Circle( float px, float py, float pr, float psp, float pysp, int pid ) {
    x = px;
    y = py;
    r = pr;
    r2 = r*r;
    id = pid;
    sp = psp;
    ysp = pysp;
  }

  void update() {
    for(int i=0; i<numCircle; i++) {
      if(i != id) {
        intersect( this, circles[i] );
      }
    }
  }

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

  void move() {
    x += sp;
    y += ysp;
    if(sp > 0) {
      if(x > width+r) {
        x = -r;
      }
    } else {
      if(x < -r) {
        x = width+r;
      }
    }
    if(ysp > 0) {
      if(y > height+r) {
        y = -r;
      }
    } else {
      if(y < -r) {
        y = height+r;
      }
    }
  }
}
```

```
void intersect( Circle cA, Circle cB )
{
  float dx = cA.x - cB.x;
  float dy = cA.y - cB.y;
  float d2 = dx*dx + dy*dy;
  float d = sqrt( d2 );

  if ( d>cA.r+cB.r || d<abs(cA.r-cB.r) ) {
    return; // no solution
  }

  //  calculate the two intersections between the two circles cA and cB, //
  //  whose coordinates are (paX, paY) and (pbX, pbY), respectively.     //

  stroke(255-dist(paX, paY, pbX, pbY)*4);
  line(paX, paY, pbX, pbY);
}
```

1. Completare la parte mancante inerente il calcolo delle intersezioni dei cerchi, per poter disegnare le corde congiungenti le coppie di punti intersezione. E' possibile basarsi sul calcolo delle coordinate delle intersezioni in un sistema di riferimento ad hoc (**"Intersezione cerchio-cerchio"**) e poi riportare il risultato nelle coordinate della finestra di Processing.
2. Rendere le corde variabili nel tempo dando diverse velocità di spostamento ai cerchi.

Exercise 4 ***(Solution on p. 15.)***

Rendere interattivo lo sketch di Exercise , facendo per esempio dipendere la quantità di spostamento dei cerchi dalla posizione lungo l'asse x del mouse.

Solutions to Exercises in this Module

Solution to Exercise (p. 8)
La variabile `mouseX` è di tipo `int`, e quindi la divisione a cui è sottoposta è di tipo intero. E' necessario fare un **type casting** da `int` a `float` mediante `(float)mouseX`.
Solution to Exercise (p. 8)

```
7
8
7
7
```

Solution to Exercise (p. 8)

```
/*

   Structure 3

   A surface filled with one hundred medium to small sized circles.
   Each circle has a different size and direction, but moves at the same slow rate.
   Display:
   A. The instantaneous intersections of the circles
   B. The aggregate intersections of the circles

   Implemented by Casey Reas <http://groupc.net>
   8 March 2004
   Processing v.68 <http://processing.org>

   modified by Pietro Polotti
   28 March, 2006
   Processing v.107 <http://processing.org>


*/

int numCircle = 100;
Circle[] circles = new Circle[numCircle];

void setup()
{
  size(800, 600);
  frameRate(50);
  for(int i=0; i<numCircle; i++) {
    circles[i] = new Circle(random(width),
      (float)height/(float)numCircle * i,
      int(random(2, 6))*10, random(-0.25, 0.25),
      random(-0.25, 0.25), i);
  }
```

```
  ellipseMode(CENTER_RADIUS);
  background(255);
}


void draw()
{
  background(255);
  stroke(0);


  for(int i=0; i<numCircle; i++) {
    circles[i].update();
  }
  for(int i=0; i<numCircle; i++) {
    circles[i].move();
  }
  for(int i=0; i<numCircle; i++) {
    circles[i].makepoint();
  }
  noFill();
}


class Circle
{
  float x, y, r, r2, sp, ysp;
  int id;

  Circle( float px, float py, float pr, float psp, float pysp, int pid ) {
    x = px;
    y = py;
    r = pr;
    r2 = r*r;
    id = pid;
    sp = psp;
    ysp = pysp;
  }

  void update() {
    for(int i=0; i<numCircle; i++) {
      if(i != id) {
        intersect( this, circles[i] );
      }
    }
  }

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

```
  void move() {
    x += sp;
    y += ysp;
    if(sp > 0) {
      if(x > width+r) {
        x = -r;
      }
    } else {
      if(x < -r) {
        x = width+r;
      }
    }
    if(ysp > 0) {
      if(y > height+r) {
        y = -r;
      }
    } else {
      if(y < -r) {
        y = height+r;
      }
    }
  }
}

void intersect( Circle cA, Circle cB )
{
  float dx = cA.x - cB.x;
  float dy = cA.y - cB.y;
  float d2 = dx*dx + dy*dy;
  float d = sqrt( d2 );

  if ( d>cA.r+cB.r || d<abs(cA.r-cB.r) ) {
    return; // no solution
  }

  float a = (cA.r2 - cB.r2 + d2) / (2*d);
  float h = sqrt( cA.r2 - a*a );
  float x2 = cA.x + a*(cB.x - cA.x)/d;
  float y2 = cA.y + a*(cB.y - cA.y)/d;

  float paX = x2 + h*(cB.y - cA.y)/d;
  float paY = y2 - h*(cB.x - cA.x)/d;
  float pbX = x2 - h*(cB.y - cA.y)/d;
  float pbY = y2 + h*(cB.x - cA.x)/d;

  stroke(255-dist(paX, paY, pbX, pbY)*4);
  line(paX, paY, pbX, pbY);
}
```

Solution to Exercise (p. 11)

```
     /*

   Structure 3

   A surface filled with one hundred medium to small sized circles.
   Each circle has a different size and direction, but moves at the same slow rate.
   Display:
   A. The instantaneous intersections of the circles
   B. The aggregate intersections of the circles

   Implemented by Casey Reas <http://groupc.net>
   8 March 2004
   Processing v.68 <http://processing.org>

   modified by Pietro Polotti
   28 March, 2006
   Processing v.107 <http://processing.org>


*/

int numCircle = 100;
Circle[] circles = new Circle[numCircle];

void setup()
{
  size(800, 600);
  frameRate(50);
  for(int i=0; i<numCircle; i++) {
    circles[i] = new Circle(random(width),
      (float)height/(float)numCircle * i,
      int(random(2, 6))*10, random(-0.25, 0.25),
      random(-0.25, 0.25), i);
  }
  ellipseMode(CENTER_RADIUS);
  background(255);
}


void draw()
{
  background(255);
  stroke(0);

  if(mousePressed){
  for(int i=0; i<numCircle; i++) {
    circles[i].sp = mouseX*random(-5, 5)/width;
```

```
  }
  }

  for(int i=0; i<numCircle; i++) {
    circles[i].update();
  }
  for(int i=0; i<numCircle; i++) {
    circles[i].move();
  }
  for(int i=0; i<numCircle; i++) {
    circles[i].makepoint();
  }
  noFill();
}


class Circle
{
  float x, y, r, r2, sp, ysp;
  int id;

  Circle( float px, float py, float pr, float psp, float pysp, int pid ) {
    x = px;
    y = py;
    r = pr;
    r2 = r*r;
    id = pid;
    sp = psp;
    ysp = pysp;
  }

  void update() {
    for(int i=0; i<numCircle; i++) {
      if(i != id) {
        intersect( this, circles[i] );
      }
    }
  }

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

  void move() {
    x += sp;
    y += ysp;
    if(sp > 0) {
      if(x > width+r) {
        x = -r;
```

```
      }
    } else {
      if(x < -r) {
        x = width+r;
      }
    }
    if(ysp > 0) {
      if(y > height+r) {
        y = -r;
      }
    } else {
      if(y < -r) {
        y = height+r;
      }
    }
  }
}

void intersect( Circle cA, Circle cB )
{
  float dx = cA.x - cB.x;
  float dy = cA.y - cB.y;
  float d2 = dx*dx + dy*dy;
  float d = sqrt( d2 );

  if ( d>cA.r+cB.r || d<abs(cA.r-cB.r) ) {
    return; // no solution
  }

  float a = (cA.r2 - cB.r2 + d2) / (2*d);
  float h = sqrt( cA.r2 - a*a );
  float x2 = cA.x + a*(cB.x - cA.x)/d;
  float y2 = cA.y + a*(cB.y - cA.y)/d;

  float paX = x2 + h*(cB.y - cA.y)/d;
  float paY = y2 - h*(cB.x - cA.x)/d;
  float pbX = x2 - h*(cB.y - cA.y)/d;
  float pbY = y2 + h*(cB.x - cA.x)/d;

  stroke(255-dist(paX, paY, pbX, pbY)*4);
  line(paX, paY, pbX, pbY);
}
```