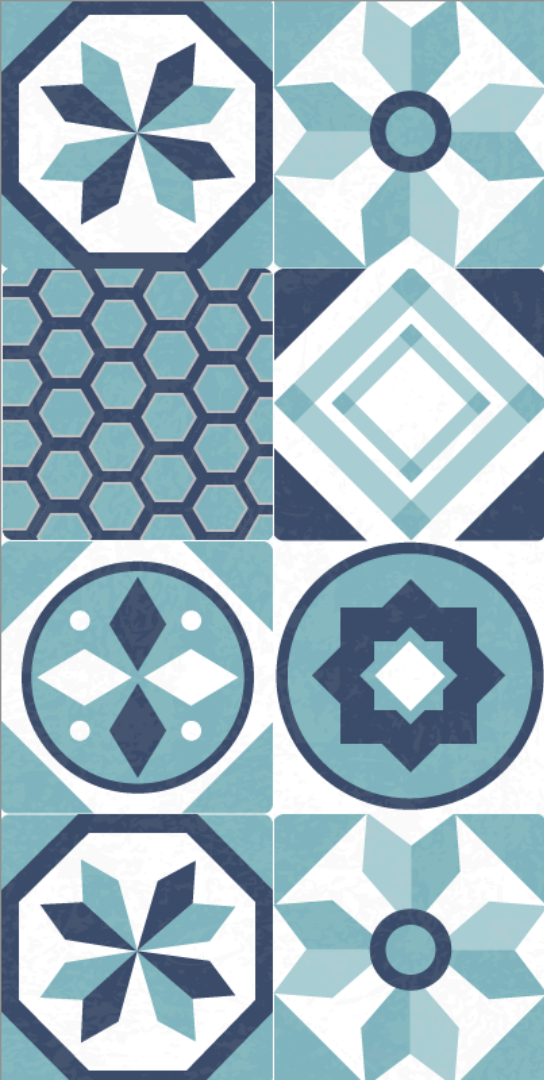
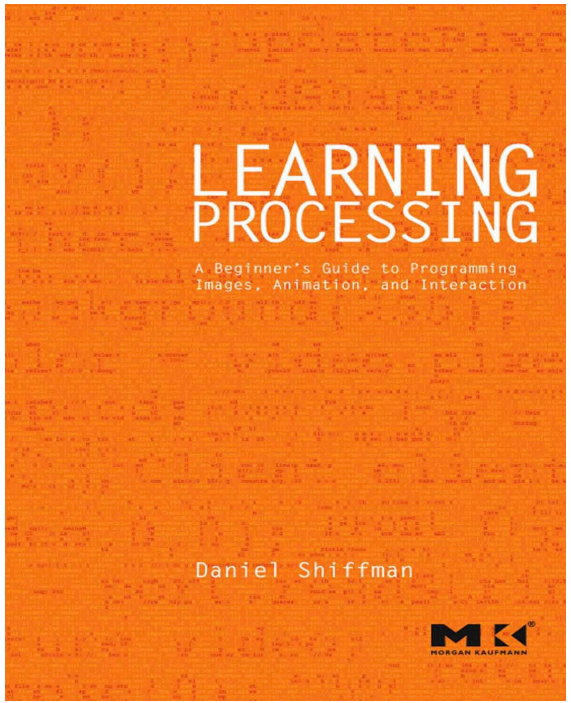


FONDAMENTI DI INFORMATICA

Alma Artis
Salvatore Rinzivillo (ISTI, CNR)

Funzioni e Oggetti

LIBRI E RIFERIMENTI



- Capitolo 7-8
- Registrarsi al sito:
  http://almaartis.rinziv.it/

Learning Processing– Second Edition
Daniel Shiffman
Available here: http://learningprocessing.com/

FUNZIONI

CODICE MODULARE

- Gli esempi che abbiamo visto finora sono abbastanza semplic e brevi
- E’ possibile identificare porzioni di codice che possono essere resi più modulari
- Un esempio di modularità già usato sono i due blocchi `setup` e `draw`
- Abbiamo usato anche funzioni predefinite da Processing: line, ellipse, point, background

PERCHÈ USARE DELLE FUNZIONI

- Modularità: permettono di dividere il codice in porzioni logiche facilmente riconoscibili
- Riusabilità: posso usare la stessa porzione di codice più volte all'interno del mio programma

ESERCIZIO

- Scrivere il programma di Zoog usando funzioni
- Quali funzioni definire?
- Come implementarle?

DEFINIZIONE DI UNA FUNZIONE

- La dichiarazione di una funzione consiste di tre parti:
  - Tipo di ritorno
  - Nome della funzione
  - Argomenti

```
returnType functionName(parameters){
  // code here
}
```

ESEMPIO

Example 7-1. Defining a function

```
void drawBlackCircle() {
  fill(0);
  ellipse(50, 50, 20, 20);
}
```

ESEMPIO - RIMBALZO

Example 7-3. Bouncing ball

```
// Declare global variables
int x = 0;
int speed = 1;

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

void draw() {
  background(255);

  // Change x by speed
  x = x + speed;

  // If its reached an edge
  // reverse speed
  if ((x > width) || (x < 0)) {
     speed = speed * -1;
  }

  // Display circle at x location
  stroke(0);
  fill(175);
  ellipse(x, 100, 32, 32);

}
```

Move the ball!

Bounce the ball!

Display the ball!

Example 7-4. Bouncing ball with functions

```
// Declare all global variables (stays the same)
int x = 0;
int speed = 1;

// Setup does not change
void setup() {
  size(200, 200);
}

void draw() {
  background(255);
  move();
  bounce();
  display();
}

// A function to move the ball
void move() {
  // Change the x location by speed
  x = x + speed;
}

// A function to bounce the ball
void bounce() {
  // If its reached an edge, reverse speed
  if ((x > width) || (x < 0)) {
    speed = speed * -1;
  }
}

// A function to display the ball
void display() {
  stroke(0);
  fill(175);
  ellipse(x, 100, 32, 32);
}
```

Instead of writing out all the code about the ball in `draw()`, I simply call three functions. How do I know the names of these functions? I made them up!

Where should functions be placed? You can define your functions anywhere in the code outside of `setup()` and `draw()`. However, the convention is to place your function definitions below `draw()`.

ARGOMENTI DI UNA FUNZIONE

- Gli argomenti sono valori che vengono passati ad una funzione
  - Ovvero sono degli input visibili solo dalla funzione

```
void drawBlackCircle(int diameter) {
  fill(0);
  ellipse(50, 50, diameter, diameter);
}

drawBlackCircle(16);  // Draw the circle with a diameter of 16
drawBlackCircle(32);  // Draw the circle with a diameter of 32
```

`diameter` is a parameter to the function `drawBlackCircle()`.

ESEMPIO

```
size(200, 200);
background(255);
int x = 100;                // x location
int y = 100;                // y location
int thesize = 64;           // size
int offset = thesize/4;     // position of wheels
relative to car

// Draw main car body (i.e., a rect)
rectMode(CENTER);
stroke(0);
fill(175);
rect(x, y, thesize, thesize/2);

// Draw four wheels relative to center
fill(0);
rect(x - offset, y - offset, offset, offset/2);
rect(x + offset, y - offset, offset, offset/2);
rect(x - offset, y + offset, offset, offset/2);
rect(x + offset, y + offset, offset, offset/2);
```

The car shape is five rectangles, one large rectangle in the center...

...and four wheels on the outside.

Figure 7-1

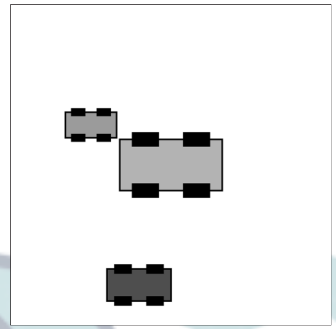
PARAMETRI DI UNA FUNZIONE

```
drawCar(80, 175, 40, color(100, 0, 100));
```

passing parameters

```
void drawCar(int x, int y, int thesize, color c) {
  // The code for the function goes here.
}
```

Figure 7-4



TIPO DI RITORNO

```
returnType functionName(parameters){
  // code here
}
```

```
void drawCar(int x, int y, int theSize, color c) {
  int offset = theSize/4;
  // Draw main car body
  rectMode(CENTER);
  stroke(200);
  fill(c);
  // Draw four wheels relative to center
  fill(200);
  rect(x - offset, y - offset, offset, offset/2);
  rect(x + offset, y - offset, offset, offset/2);
  rect(x - offset, y + offset, offset, offset/2);
  rect(x + offset, y + offset, offset, offset/2);
}
```

RITORNO DI UN VALORE

```
int sum(int a, int b, int c) {

  int total = a + b + c;
  return total;
}
```

This function, which adds three numbers together, has a return type: `int`.

A return statement is required! A function with a return type must always return a value of that type.

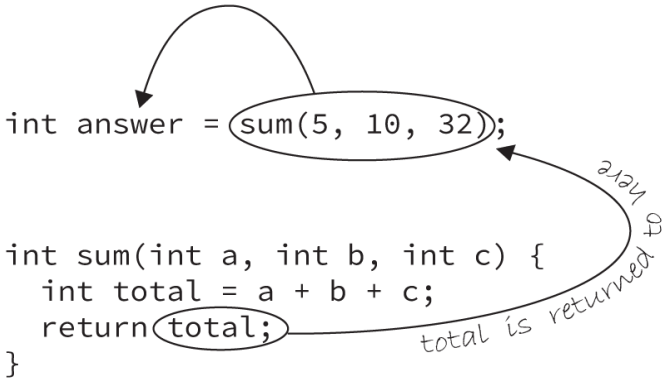


Figure 7-5

ESEMPIO – FUNZIONE DISTANZA

```
float distance(float x1, float y1, float x2, float y2) {
  float dx = x1 - x2;
  float dy = y1 - y2;
  float d = sqrt(dx*dx + dy*dy);
  return d;
}
```

Our version of Processing's `dist()` function.

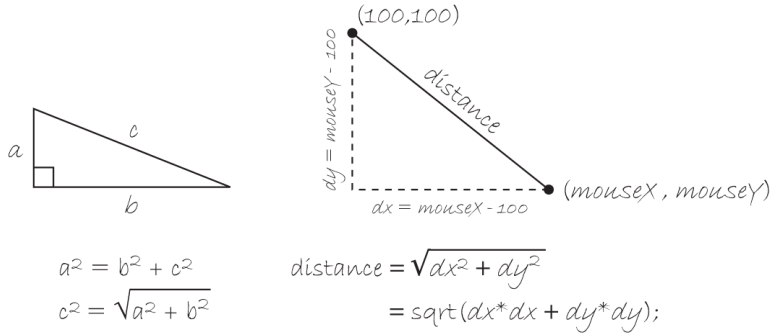


Figure 7-6

ROLLOVER CON UN CERCHIO

Example 7-5. Using a function that returns a value, distance

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


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

  float d = distance(width/2, height/2, mouseX, mouseY);

  fill(d*3, d*2, d);
  ellipseMode(CENTER);
  ellipse(width/2, height/2, 100, 100);
}

float distance(float x1, float y1, float x2, float y2) {
  float dx = x1 - x2;
  float dy = y1 - y2;
  float d = sqrt(dx*dx + dy*dy);
  return d;
}
```

The result of the `distance()` function is used to color a circle. I could have used the built-in function `dist()` instead, but I am demonstrating how how to define a function that returns a value.

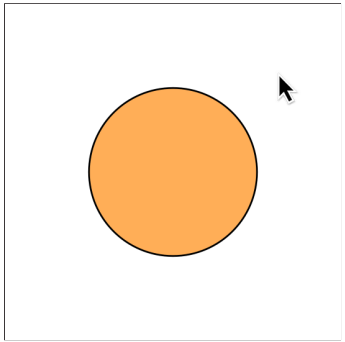


Figure 7-7

ESERCIZIO

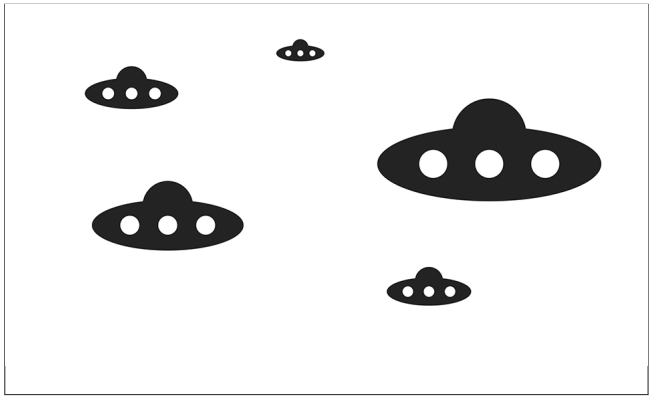
- Riorganizzazione del codice di Zoog
- V. Libro Esempio 7-6

ESERCIZIO 7-10

example code.

```
void setup() {
  size(640, 360);
}

void draw() {
  background(255);
  spaceShip(482, 159, 223);
  spaceShip(126, 89, 93);
  spaceShip(422, 286, 84);
  spaceShip(294, 49, 48);
  spaceShip(162, 220, 151);
}
```



Note the use of only three arguments: x, y, and size. However, you might consider adding parameters for color, number of windows, and more.