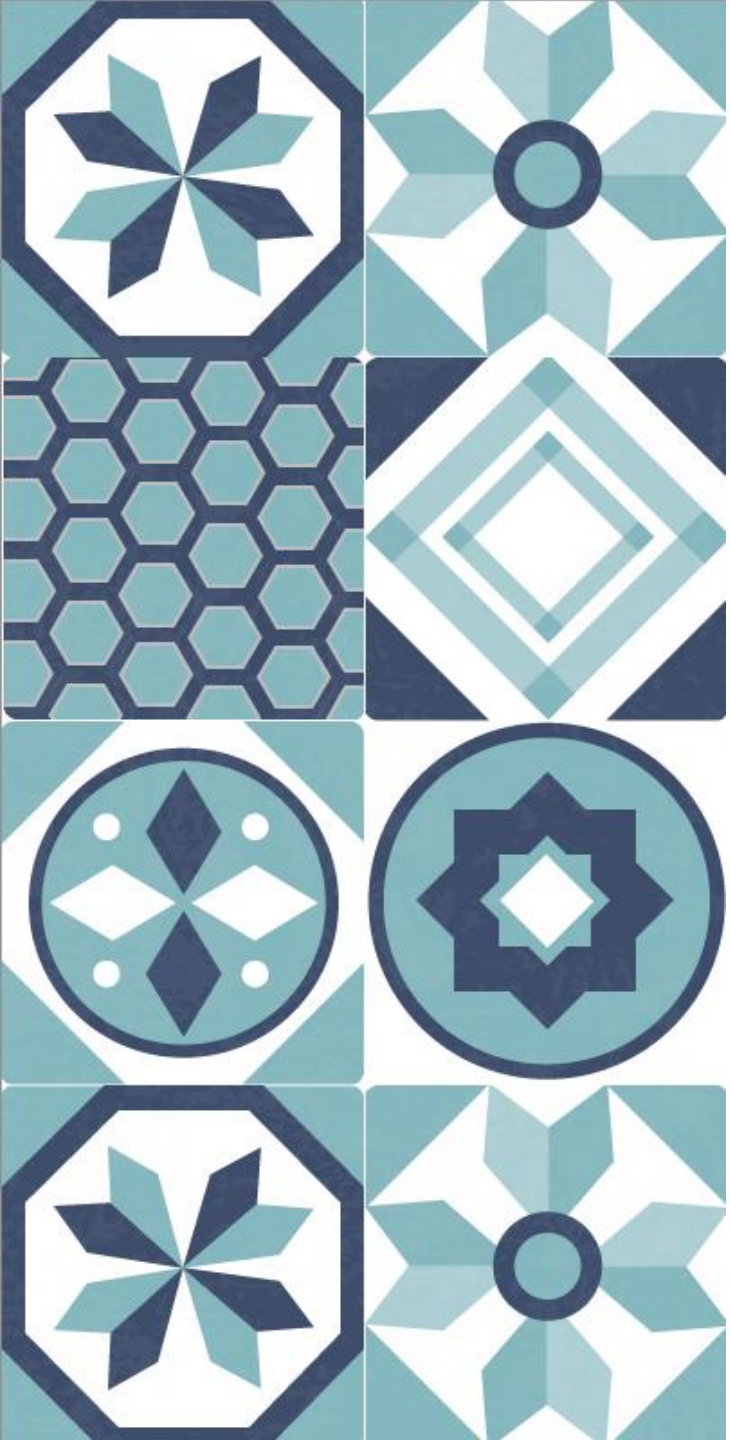


FONDAMENTI DI INFORMATICA

Alma Artis
Francesca Pratesi (ISTI, CNR)

Processing – Funzioni avanzate, Oggetti, Array, Immagini

ESERCIZIO – RIMBALZO DI UNA PALLA

- Creare uno sketch che disegni una palla (un cerchio) che si muove sullo schermo in linea retta
- Quando tocca il bordo della finestra, inverte la direzione del moto e torna indietro

ESERCIZIO

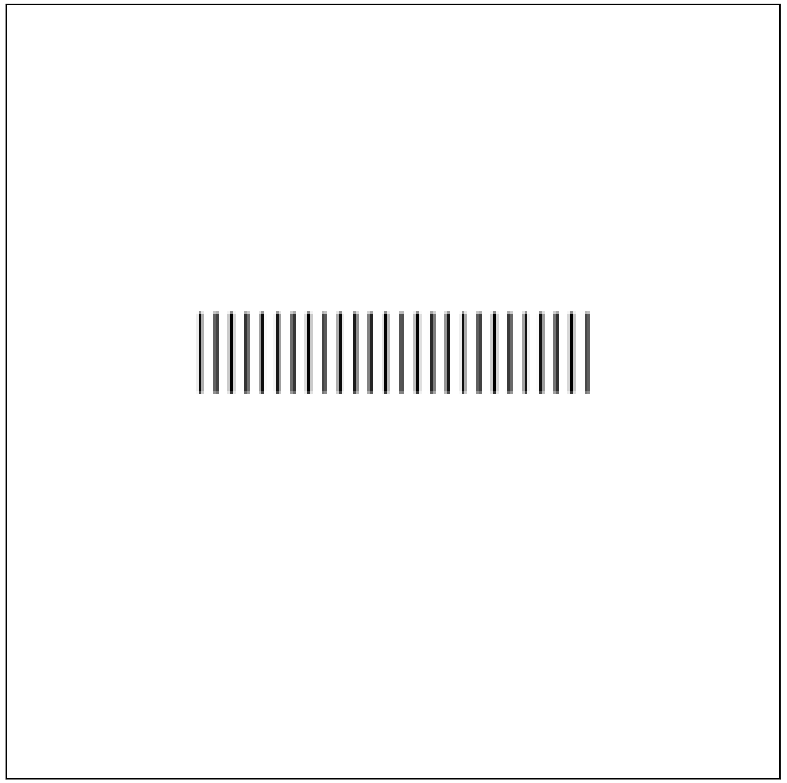


Figure 6-4

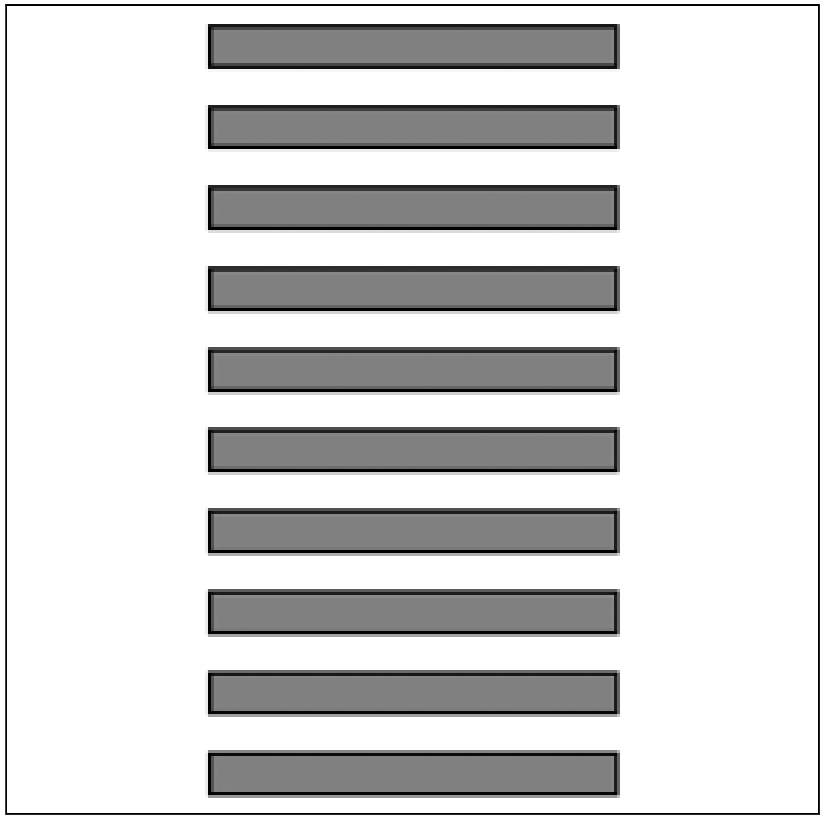
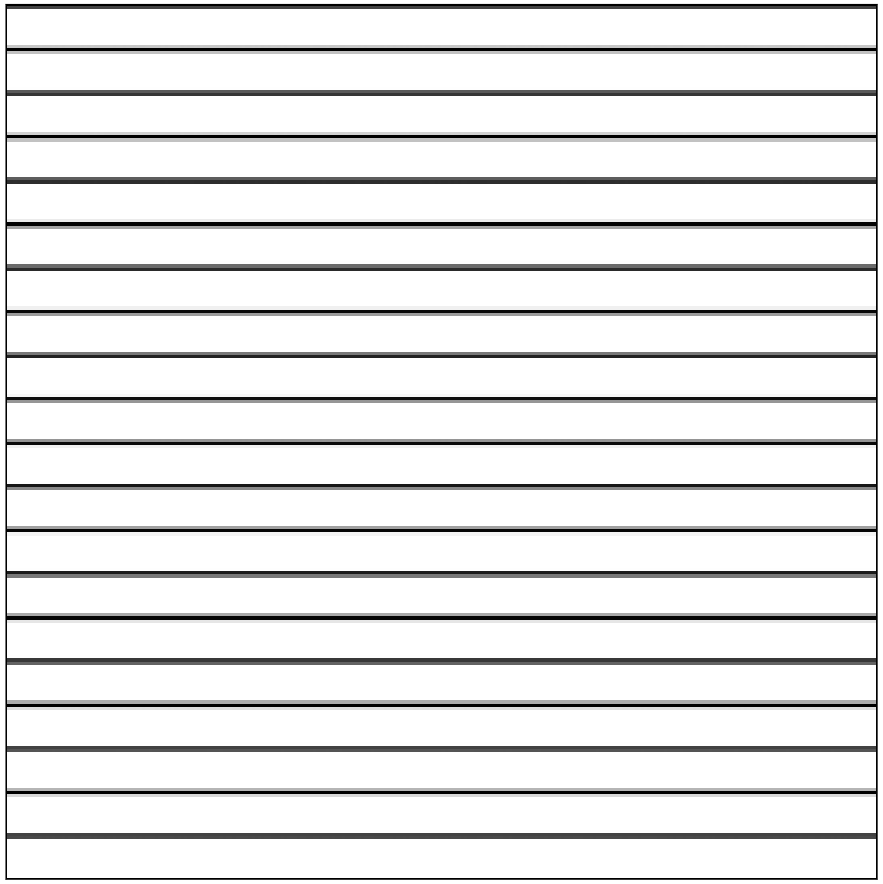
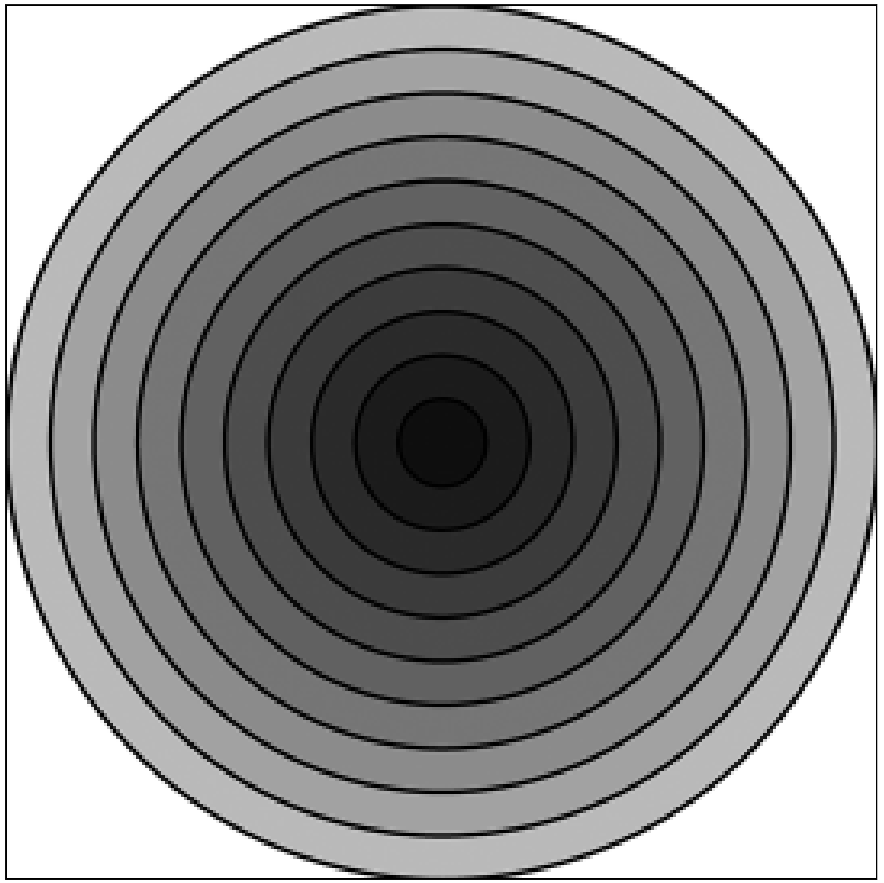


Figure 6-5

ESERCIZIO 6-1





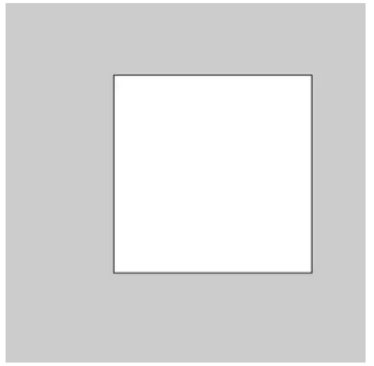
PERSONALIZZAZIONE

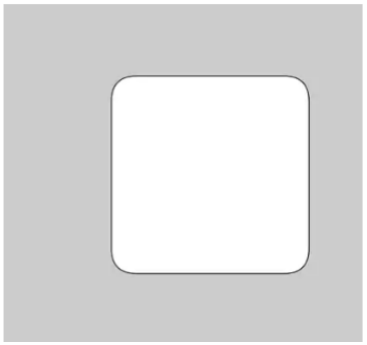
PERSONALIZZARE UN RETTANGOLO

- È possibile disegnare anche rettangoli con angoli arrotondati
- Basta aggiungere parametri alla primitiva
- Un quinto parametro rappresenterà quanto arrotondare gli angoli
- È possible dare valori diversi ai quattro angoli, in questo caso saranno necessari anche un sesto, settimo e ottavo parametro

PERSONALIZZARE UN RETTANGOLO - ESEMPIO

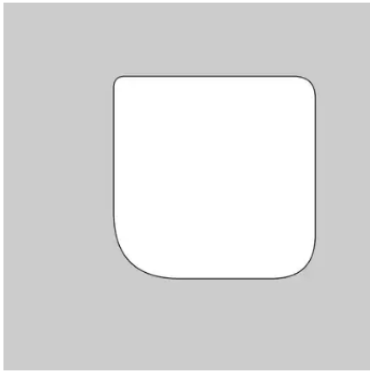
```
rect(120, 80, 220, 220);
```





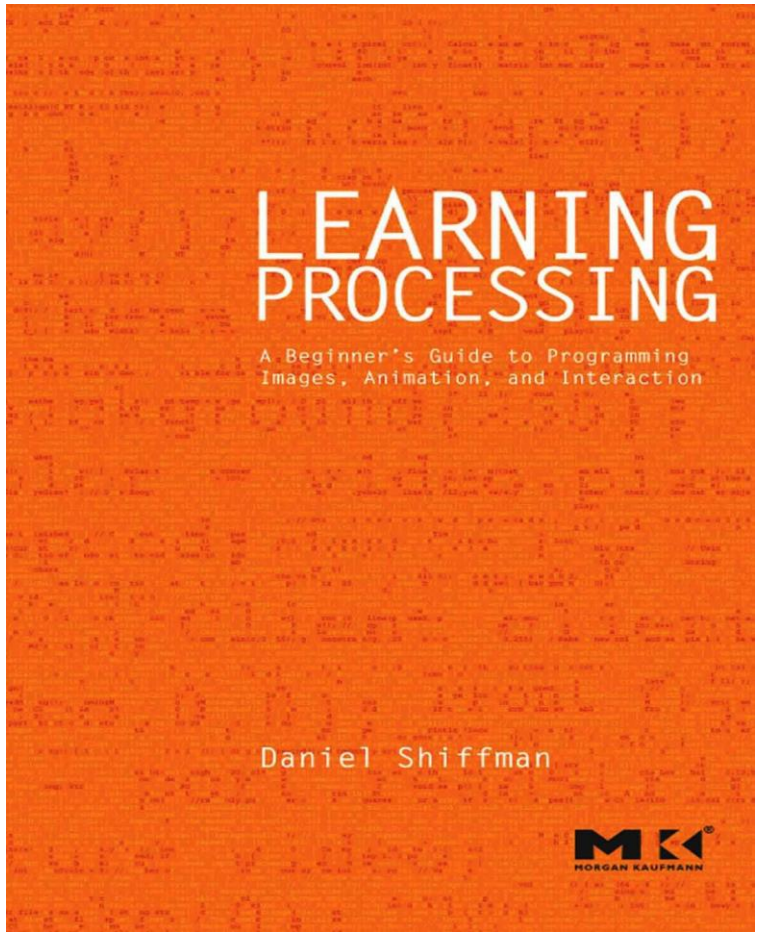
```
rect(120, 80, 220, 220, 28);
```

```
rect(120, 80, 220, 220, 12, 24, 48, 72);
```



FUNZIONI DI DISEGNO

LIBRI E RIFERIMENTI



- Capitolo 14

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

TRASFORMAZIONI AFFINI

- La posizione di un punto sullo schermo è gestita tramite coordinate del canvas
- Le coordinate fanno riferimento ad un sistema di riferimento con la posizione (0,0) posizionata nell'angolo in alto a destra del canvas
- Per ottenere una forma alle coordinate (20,20) possiamo utilizzare la seguente istruzione

```
rect(20,20,20,40)
```

- In alternativa, possiamo spostare la posizione del sistema di riferimento attraverso una operazione chiamata trasformazione affine
- Le trasformazioni sono operazioni matematiche che soddisfano alcune proprietà: preservano distanza, forma e superfici

TRASFORMAZIONI

- Tre tipi di trasformazioni di base
  - Translazione
  - Rotazione
  - Scala
- Altre trasformazioni più complesse:
  - shear
  - Trasformazioni tramite matrice

TRANSLAZIONE

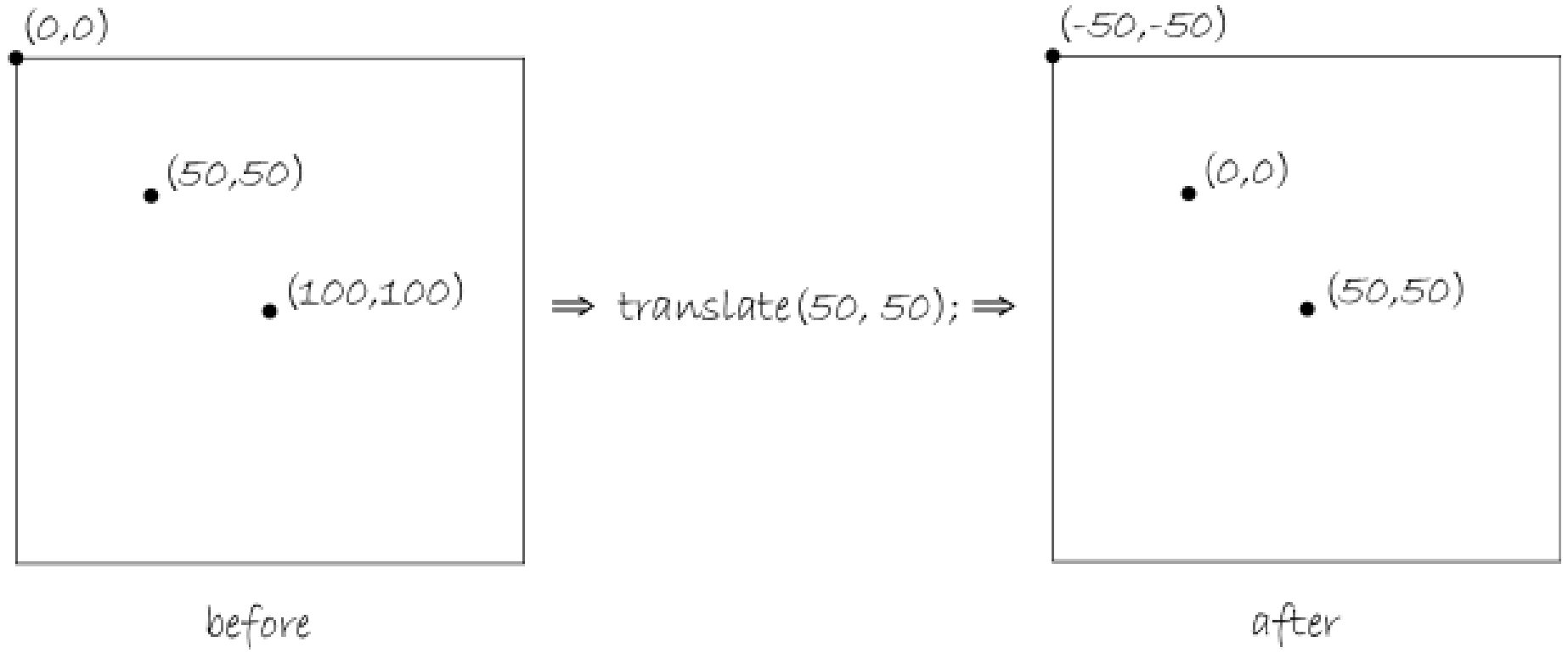


Figure 14-3

ESEMPIO 14-2

Example 14-2. Multiple translations

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

void draw() {
  background(255);
  stroke(0);
  fill(175);

  // Grab mouse coordinates, constrained to window
  int mx = constrain(mouseX, 0, width);
  int my = constrain(mouseY, 0, height);


  translate(mx, my);
  ellipse(0, 0, 8, 8);

  translate(100, 0);
  ellipse(0, 0, 8, 8);

  translate(0, 100);
  ellipse(0, 0, 8, 8);

  translate(-100, 0);
  ellipse(0, 0, 8, 8);
}
```

Translate to the mouse location.

Translate 100 pixels to the right.

Translate 100 pixels down.

Translate 100 pixels left.

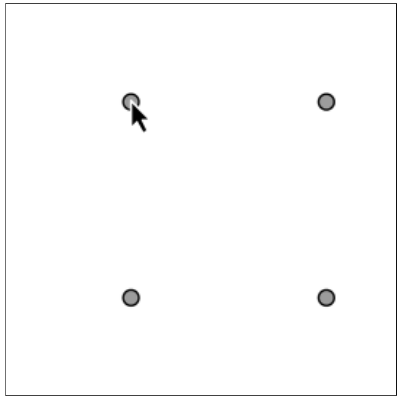
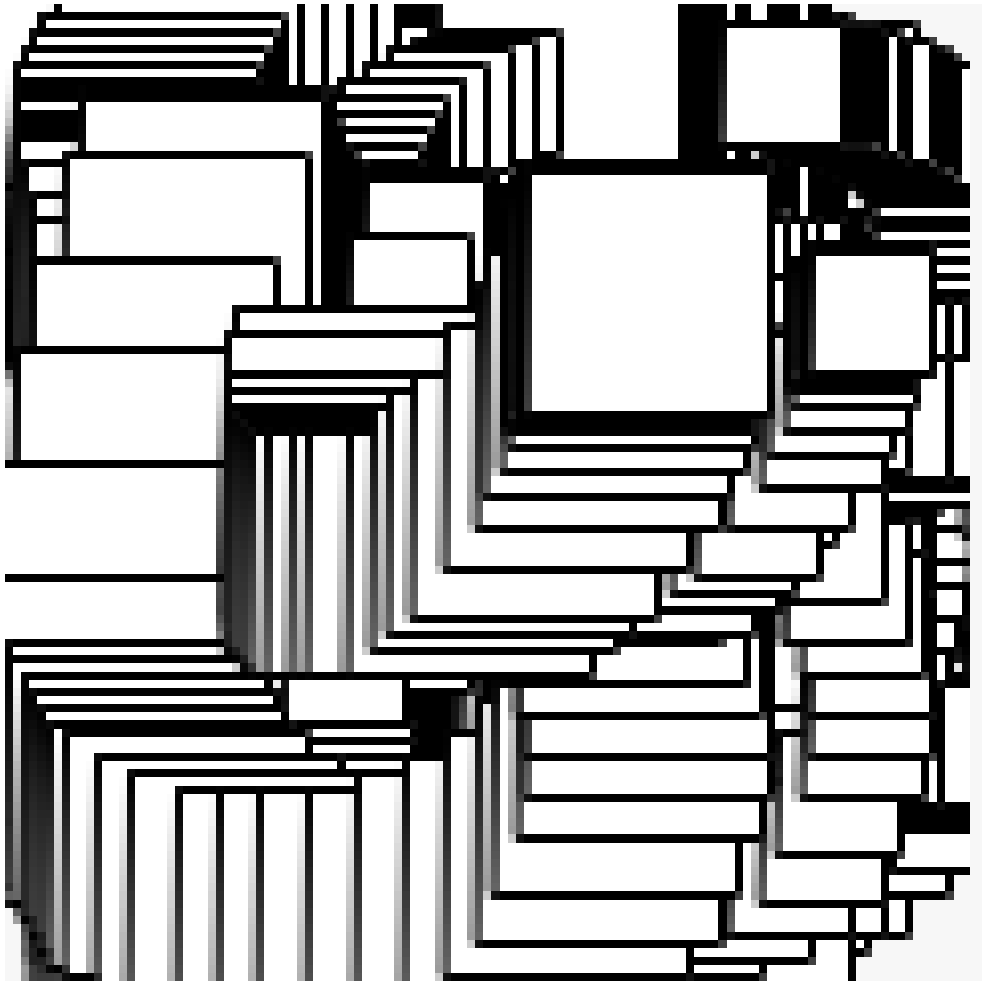


Figure 14-4

ESEMPIO 6-2:CONCATENAZIONI

```
void setup() {
  size(120, 120);
}

void draw() {
  translate(mouseX, mouseY);
  rect(0, 0, 30, 30);
  translate(35, 10);
  rect(0, 0, 15, 15);
}
```

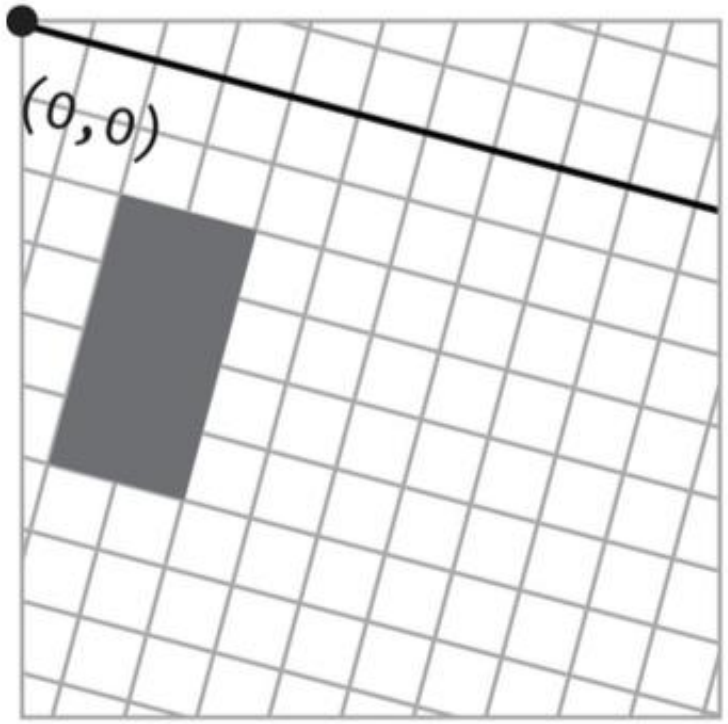


CONCATENAZIONI DI TRASLAZIONE

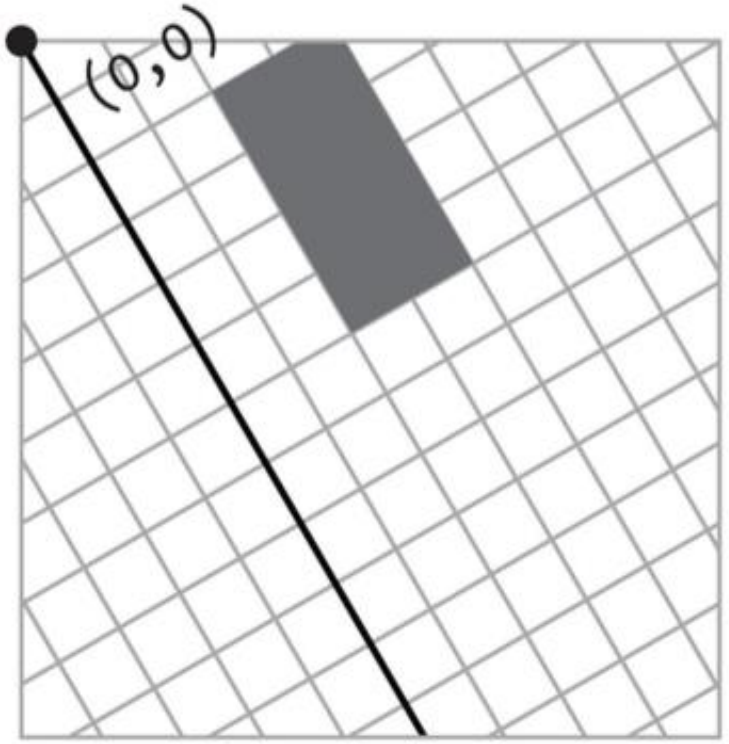
- Translate sposta quindi l'origine del nostro sistema di riferimento, e quindi di quello che stiamo per disegnare
- Gli effetti sono cumulativi:
  - `translate(10,0); translate(40,50)`
  - è lo stesso che scrivere
  - `translate(50,50)`
- Gli effetti si resettano al draw successivo

ROTAZIONE

```
rotate(PI/12.0);
rect(20, 20, 20, 40);
```




```
rotate(-PI/3);
rect(20, 20, 20, 40);
```



ROTAZIONE (2)

- Rotate ruota una forma di un angolo specificato come parametro.
- L'angolo deve essere specificato in radianti (valori da 0 a 2*π) o deve essere convertito in radiant con la funzione radians().
- Gli oggetti sono ruotati sulla loro posizione relativa di origine.
- Un numero positivo ruota la forma in senso orario.
- Le trasformazioni si applicano a tutto quello che viene disegnato dopo.
- Gli effetti si sommano
  - Es: rotate(HALF_PI) + rotate(HALF_PI) = rotate(PI)
- Tutte le trasformazioni si resettano all'inizio della draw().

ESEMPIO 14-5: ROTAZIONI ATTORNO AD UN CENTRO

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

void draw() {
  background(255);
  stroke(0);
  fill(175);
  // trasla l'origine al centro
  translate(width/2, height/2);
  // map trasforma il valore mouseX, che varia tra 0 e width,
in un range tra 0 e 6.28, cioè due volte pi Greco (TWO_PI è
una costante matematica)
  float theta = map(mouseX, 0, width, 0, TWO_PI);
  // ruota dell'angolo theta
  rotate(theta);
  rectMode(CENTER);
  rect(0,0,100,100);
}
```

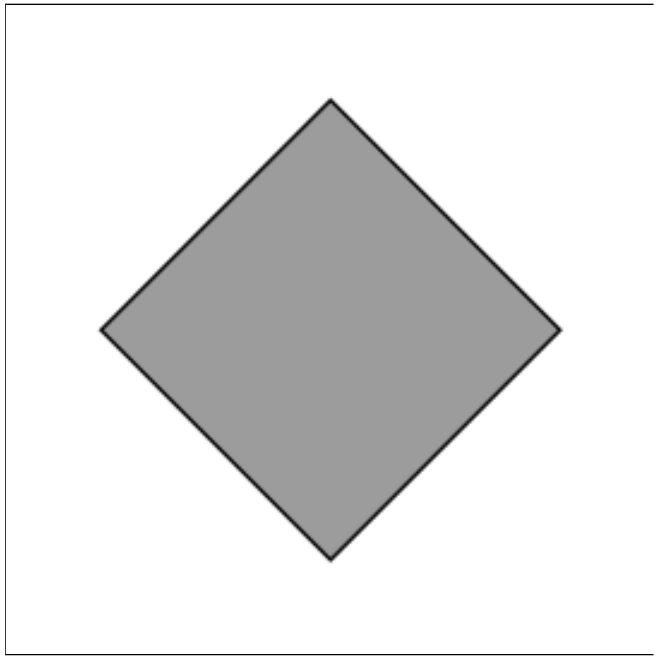
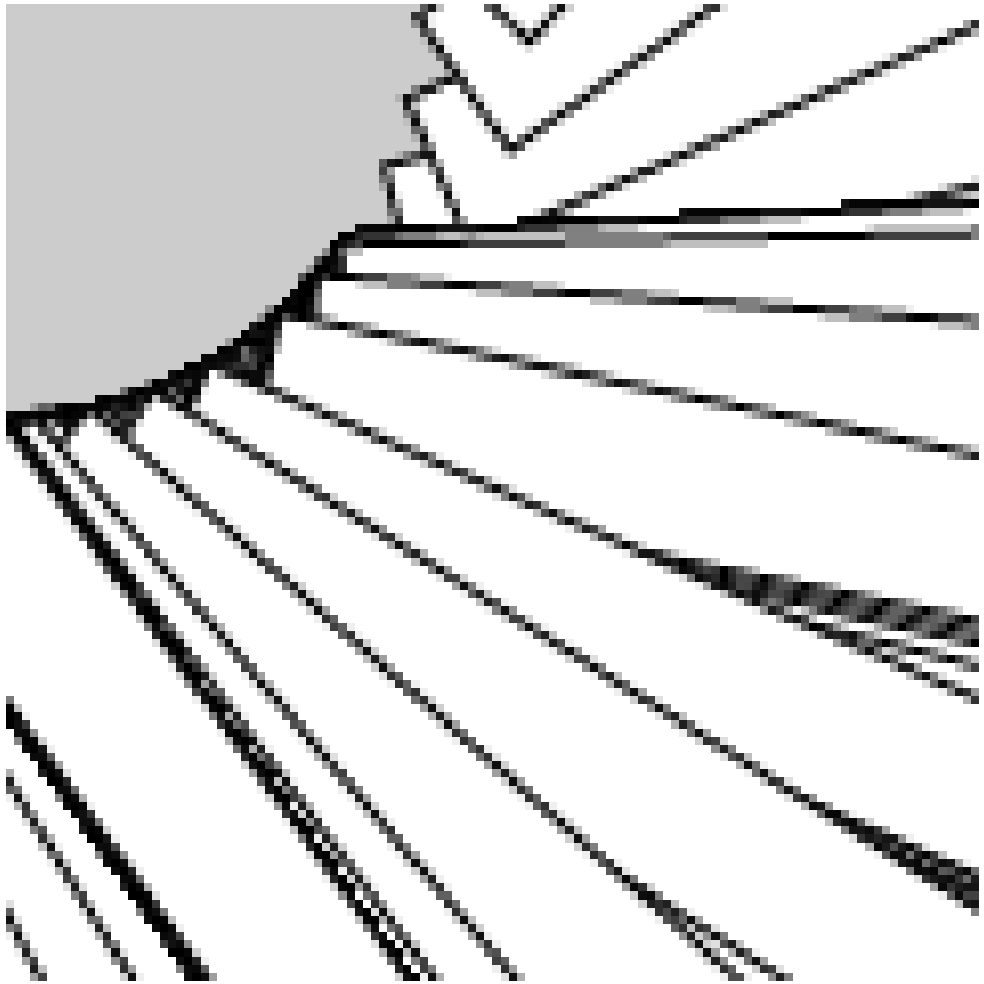


Figure 14-15

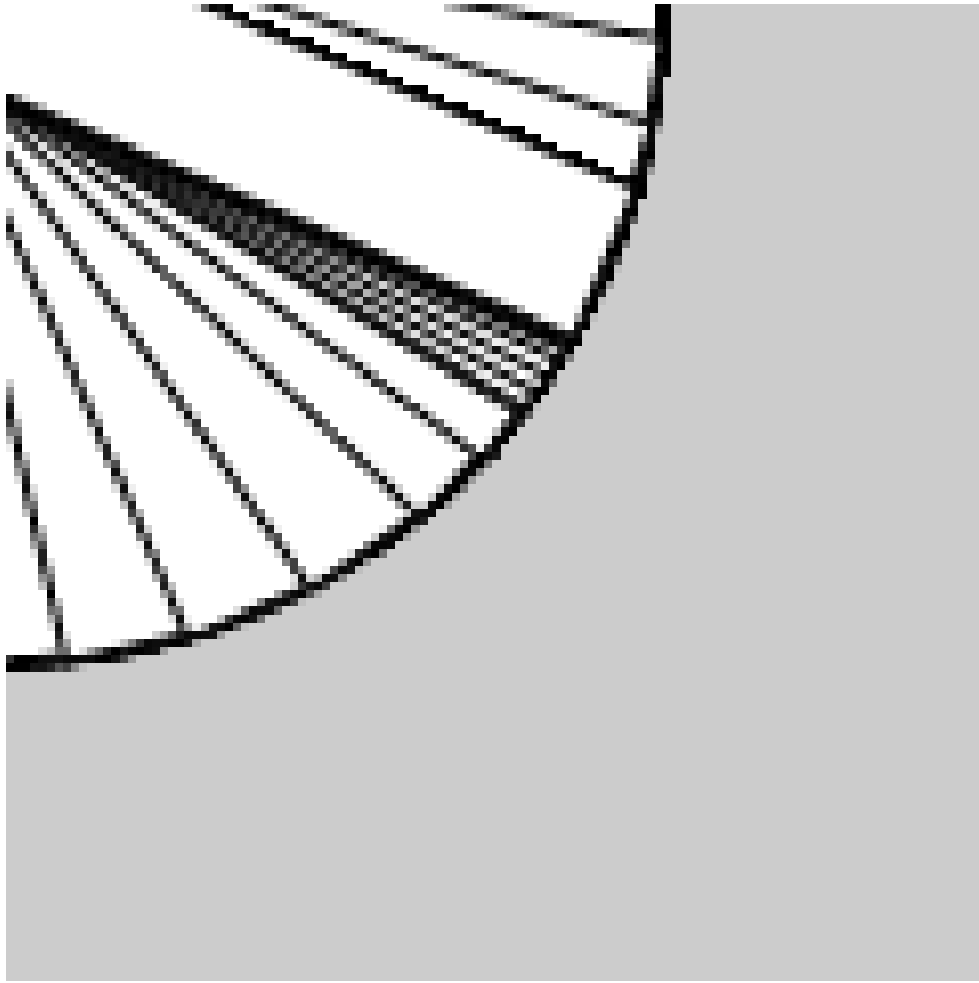
ESEMPIO : ROTAZIONE INTORNO ALL'ORIGINE

```
int x = 100;
int y = 100;
float t = 1.0;
void setup() {
  size(400, 400);
  background(204);
}

void draw() {
  translate(x, y);
  rotate(t);
  t=t+0.01;
  rectMode(CENTER);
  rect(0, 0, 160, 20);
 }
```



ESEMPIO : ROTAZIONE INTORNO AL CENTRO

```
int x = 100;
int y = 100;
float t = 1.0;
void setup() {
  size(400, 400);
  background(204);
}

void draw() {
  translate(x, y);
  //rotate(mouseX / 100.0);
  rotate(t);
  t=t+0.01;
  rect(0, 0, 160, 20);
}
```

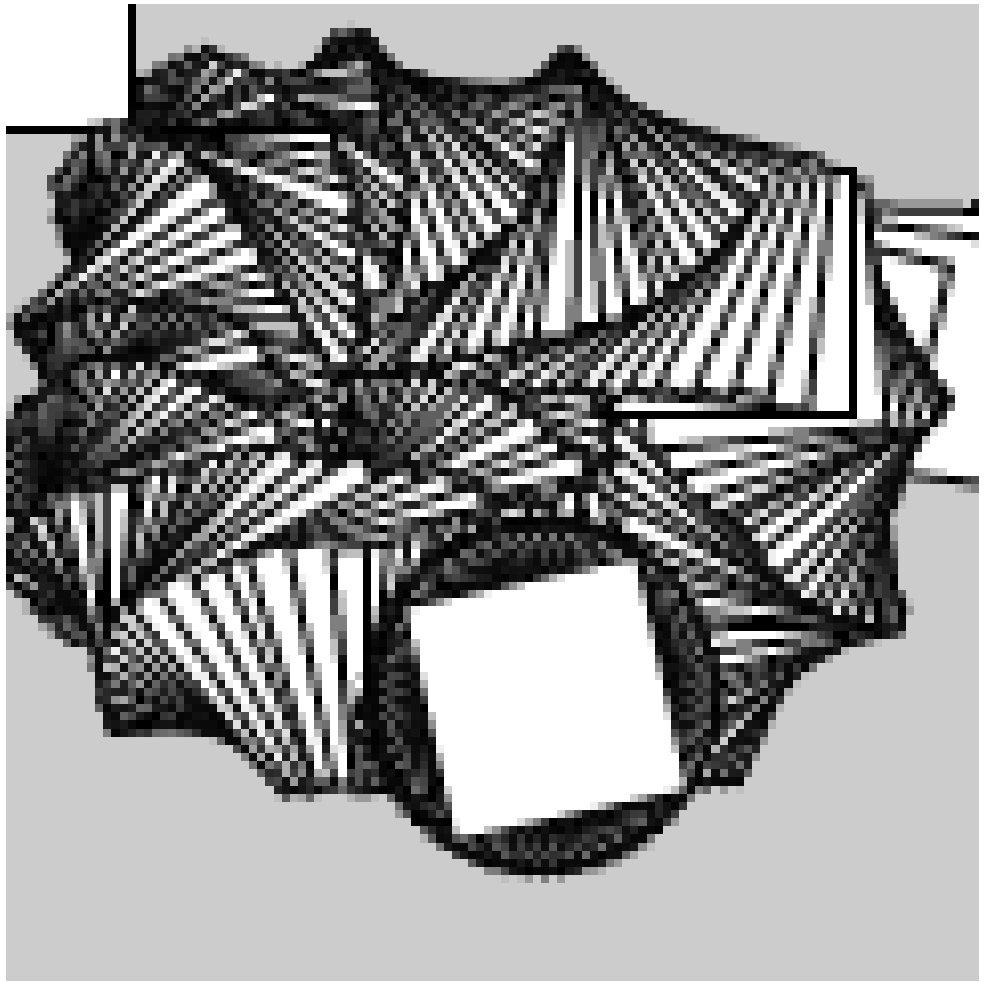


COMBINAZIONI DI TRASFORMAZIONI

- Tutte le rotazioni fanno riferimento al centro del sistema di riferimento
- Per avere una rotazione intorno ad un punto qualsiasi è necessario utilizzare una combinazioni di trasformazioni
  - Ad esempio: una translazione verso il nuovo centro di rotazione e poi una rotazione

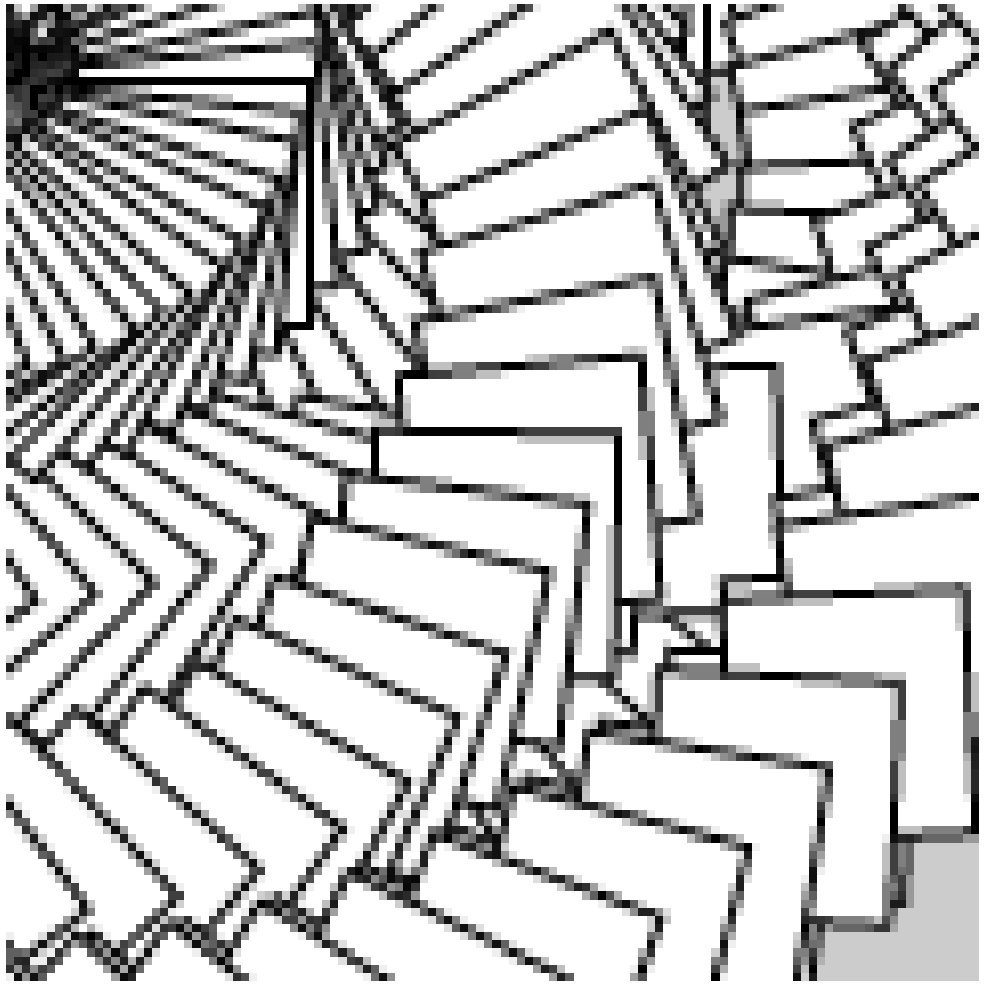
ESEMPIO : TRANSLAZIONE E ROTAZIONE

```
float angle = 0.0;

void setup() {
  size(820, 820);
  background(204);
}

void draw() {
  translate(mouseX, mouseY);
  rotate(angle);
  rect(-15, -15, 30, 30);
  angle += 0.1;
}
```



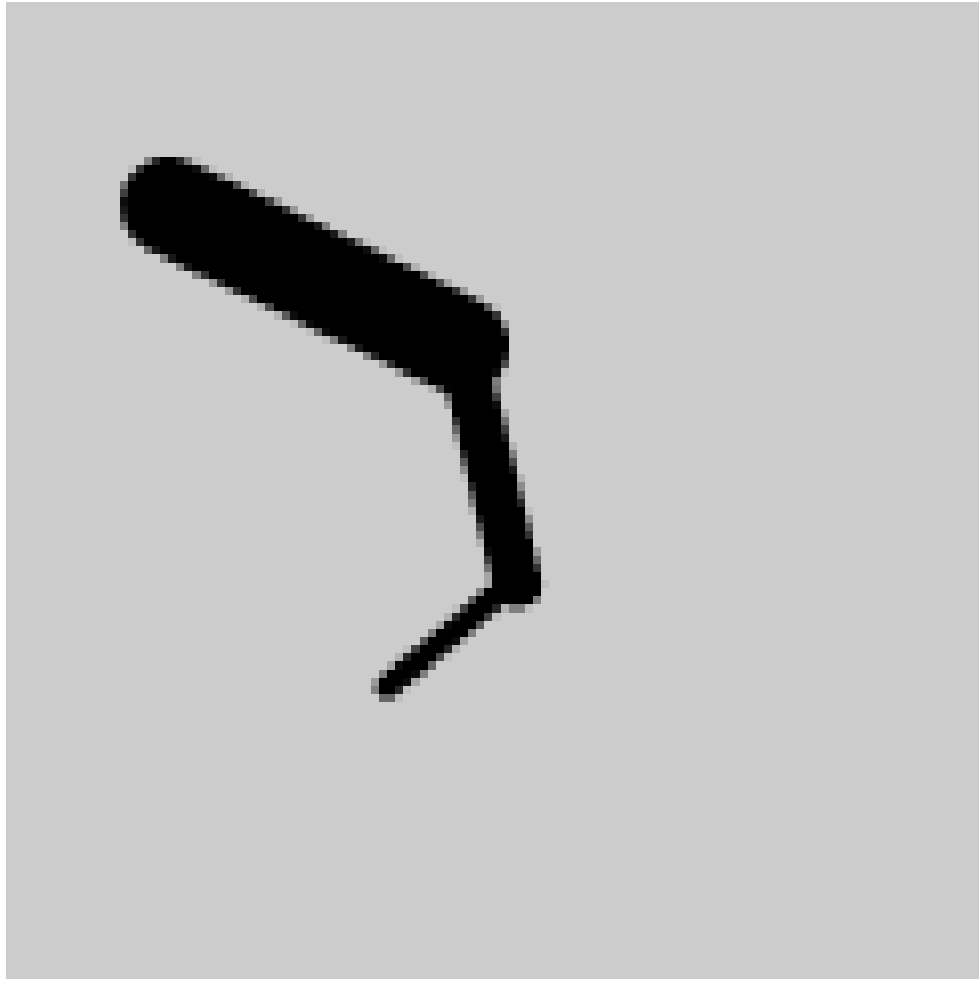
ESEMPIO: ROTAZIONE E TRASLAZIONE

```
float angle = 0.0;

void setup() {
  size(820, 820);
  background(204);
}

void draw() {
  // attenzione all'ordine delle trasformazioni!!!
  rotate(angle);
  translate(mouseX, mouseY);

  rect(-15, -15, 30, 30);
  angle += 0.1;
}
```



ESEMPIO 6-7: BRACCIO ARTICOLATO

```
float angle = 0.0;
float angleDirection = 1;
float speed = 0.005;

void setup() {
  size(120, 120);
}

void draw() {
  background(204);
  translate(20, 25);  // mi muovo alla posizione di partenza
  rotate(angle);
  strokeWeight(12);
  line(0, 0, 40, 0); // disegno la prima parte del braccio, la più spessa
  translate(40, 0);   // mi muovo alla giunzione
  rotate(angle * 2.0); // aumento la rotazione
  strokeWeight(6); // diminuisco lo spessore della linea
  line(0, 0, 30, 0); // disegno la seconda parte del braccio
  translate(30, 0);   // mi muovo alla prossima giunzione
  rotate(angle * 2.5);
  strokeWeight(3);
  line(0, 0, 20, 0);

  angle += speed * angleDirection;
  if ((angle > QUARTER_PI) || (angle < 0)) {
    angleDirection *= -1;
  }
}
```

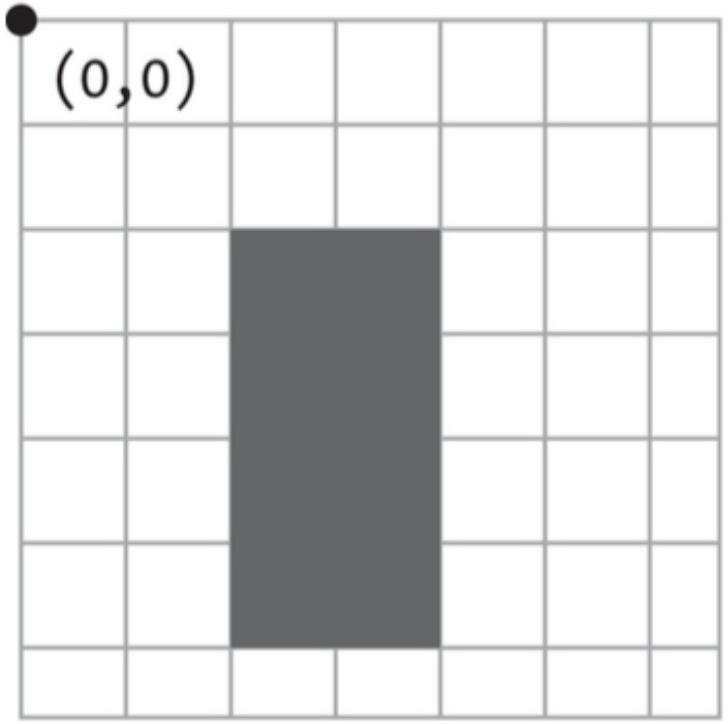


SCALA

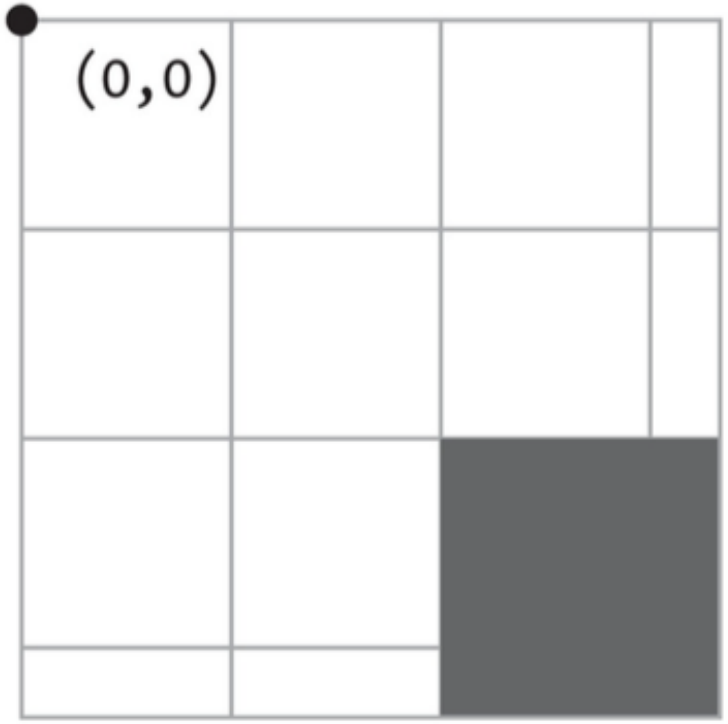
- La trasformazione di scala estende o contrae gli assi del sistema di riferimento
- Tutto quello che viene disegnato sul nuovo sistema di riferimento aumenta o diminuisce le sue dimensioni
- Il fattore di scala si indica con un numero decimale
  - 1.5 corrisponde a un scala del 150%
  - 3.0 ad una scala del 300%
  - 0.5 ad una scala del 50%

SCALA

```
scale(1.5);
rect(20, 20, 20, 40);
```

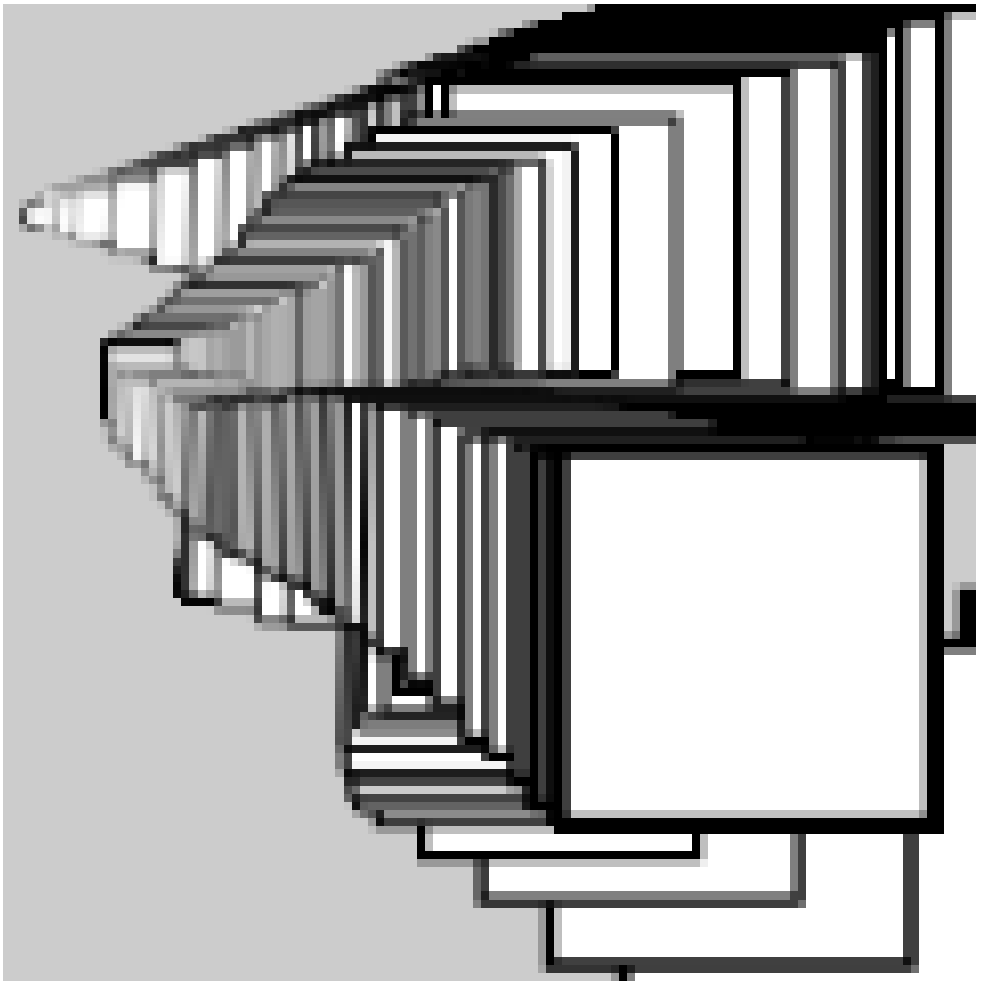



```
scale(3);
rect(20, 20, 20, 40);
```



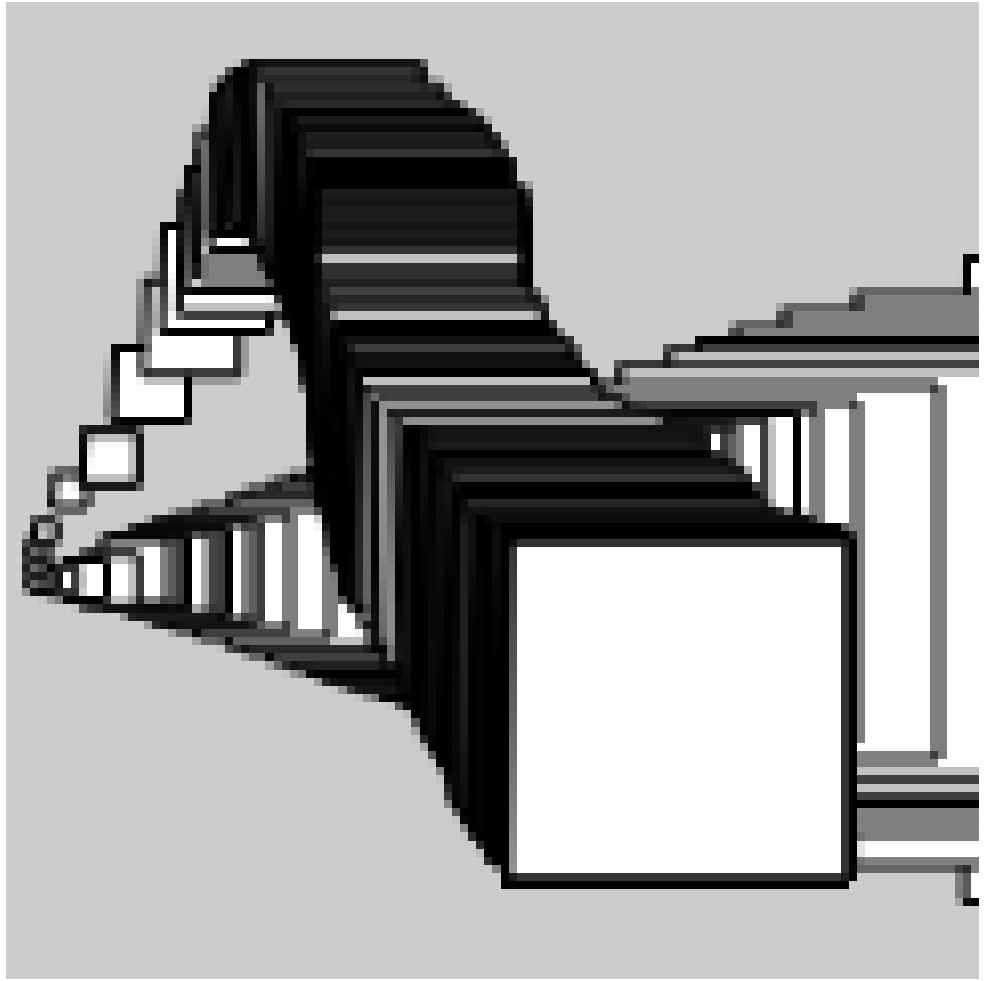
ESEMPIO: SCALA

```
void setup() {
  size(600, 600);
  background(204);
}

void draw() {
  translate(mouseX, mouseY);
  scale(mouseX / 60.0);
  rect(-15, -15, 30, 30);
}
```



ESEMPIO 6-9: SCALA DEL TRATTO

```
function setup() {
  size(600, 600);
  background(204);
}

function draw() {
  translate(mouseX, mouseY);
  var scalar = mouseX / 60.0;
  scale(scalar);
  // il valore per cui scaliamo è lo stesso, ma lo usiamo anche per modificare lo spessore della linea
  strokeWeight(1.0 / scalar);
  rect(-15, -15, 30, 30);
}
```



ISOLAMENTO DELLE TRASFORMAZIONI

- Le trasformazioni si sommano in sequenza
- Per isolare l'effetto di alcune trasformazioni si possono utilizzare le funzioni pushMatrix() e popMatrix()
- pushMatrix() (spingere) salva lo stato corrente del sistema di riferimento
  - Dopo questo comando posso modificare lo stato con nuove trasformazioni
- popMatrix() (tirare) ripristina l'ultimo stato del sistema
  - Questo comando mi permette di tornare alla configurazione relativa all'ultimo push()
- pushMatrix() e popMatrix() seguono una logica LIFO (Last In First Out)

MODALITÀ LIFO E FIFO

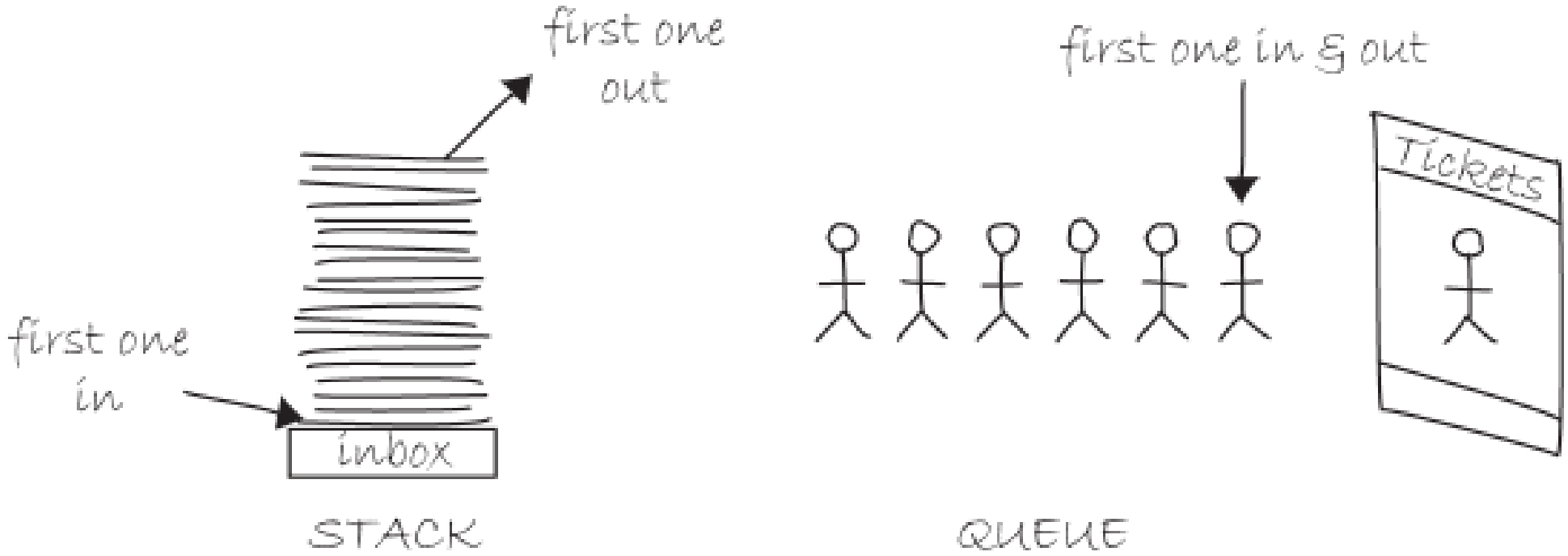


Figure 14-28

ESEMPIO 14-18: SISTEMA SOLARE

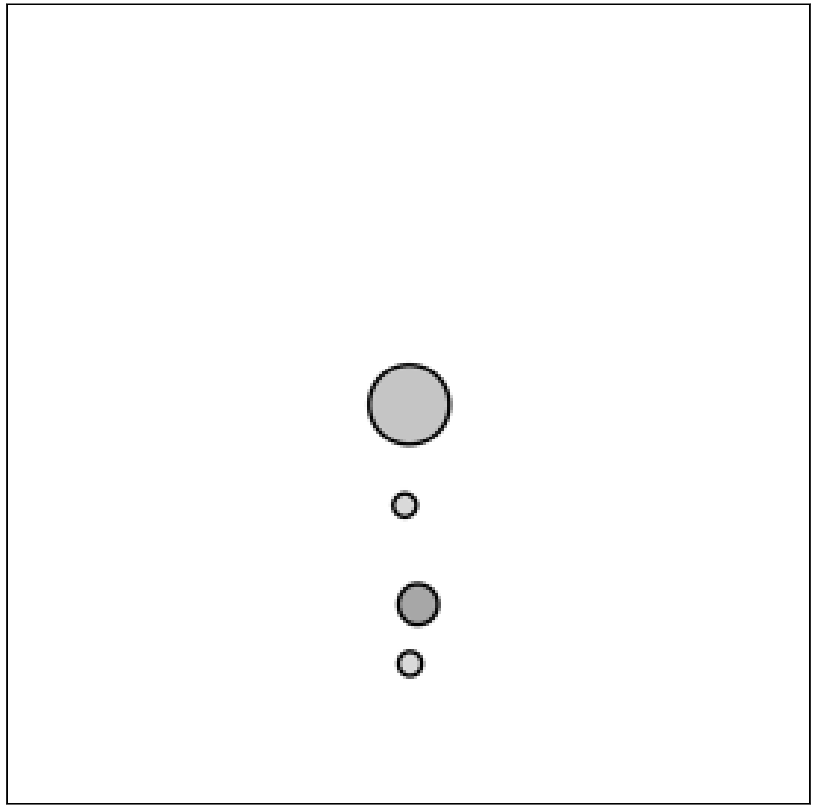
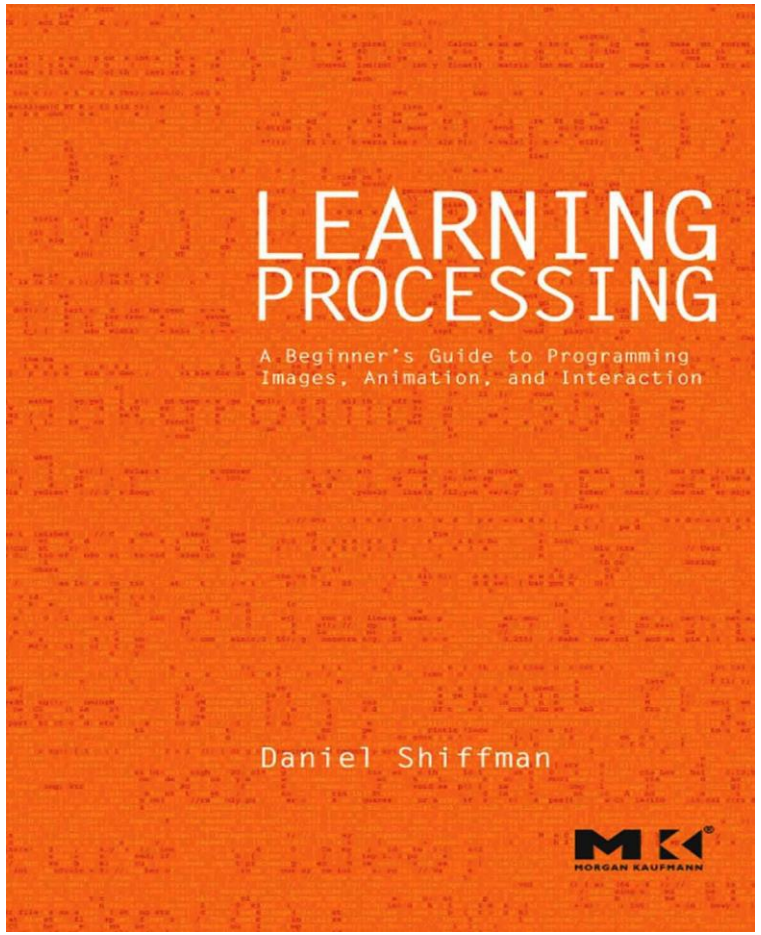


Figure 14-29

OGGETTI

LIBRI E RIFERIMENTI



- Capitolo 8

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

OGGETTI

- Fino abbiamo visto:
  - Variabili, condizionali, cicli, funzioni
- La programmazione ad oggetti mette insieme tutti questi elementi
- Cambia il modo di strutturare gli algoritmi che implementiamo
- Esempio (Essere Umano)
  - Dati
    - Altezza, peso, colore occhi, colore dei capelli, gender
  - Funzioni
    - Sveglia, Alzati, Mangia, Cammina, ecc.
- Distinzione tra Classe e Oggetto

ESERCIZIO SKETCH AUTO (FUNZIONI)

- Dati (variabili globali)
  - Colore, posizione x, posizione y, velocità
- Setup
  - Scegli un colore
  - Scegli una posizione
  - Scegli una velocità
- Draw
  - Riempi lo sfondo
  - Disegna la macchina alla posizione
  - Modifica la posizione in base alla velocità

AUTO CON FUNZIONI

```
color c;
float xpos;
float ypos;
float xspeed;

void setup(){
  size(400,400);
  c=color(255);
  xpos=width/2;
  ypos=height/2;
  xspeed=1;
}

void draw(){
 background(0);
 display();
 drive();
}
```

```
void display(){
  rectMode(CENTER);
  fill(c);
  rect(xpos,ypos,20,10);
}

void drive(){
  xpos=xpos+xspeed;
  if(xpos > width) xpos = 0;
}
```

CLASSI

- Se l'oggetto che stiamo modellando ha delle caratteristiche o delle funzionalità che vorremmo astrarre
  - (per esempio perché è più comodo sul momento…
  - o perché prevediamo di riusarlo il futuro)
- allora può convenire pensare di creare una classe per quell'oggetto
- Esempio: vogliamo definire una serie di automobili, ognuna che si muove orizzontalmente nello spazio, ma ad altezze diverse della finestra

CLASSI: DI CHE COSA HO BISOGNO?

- Dati
  - Le caratteristiche dell'oggetto
- Costruttore
  - Per creare l'oggetto
  - Qua inizializzerò i suoi dati
- Funzionalità
  - Le funzioni che potrò usare per quell'oggetto

CREAZIONE DI UNA CLASSE

```
// Simple non OOP Car

color c;
float xpos;
float ypos
float xspeed;

void setup() {
  size(200, 200);
  c = color(255);
  xpos = width/2;
  ypos = height/2;
  xspeed = 1;
}

void draw () {
  background(0);
  display();
  drive();
}

void display () {
  rectMode(CENTER);
  fill(c);
  rect(xpos, ypos, 20, 10);
}

void drive() {
  xpos = xpos + xspeed;
  if (xpos > width) {
    xpos = 0;
  }
}
```

```
class Car {

  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car() {
    c = color(255);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  void display () {
    rectMode(CENTER);
    fill(c);
    rect(xpos, ypos, 20, 10);
  }

  void drive() {
    xpos = xpos + xspeed;
    if (xpos > width) {
      xpos = 0;
    }
  }
```

The class name

Data

Constructor

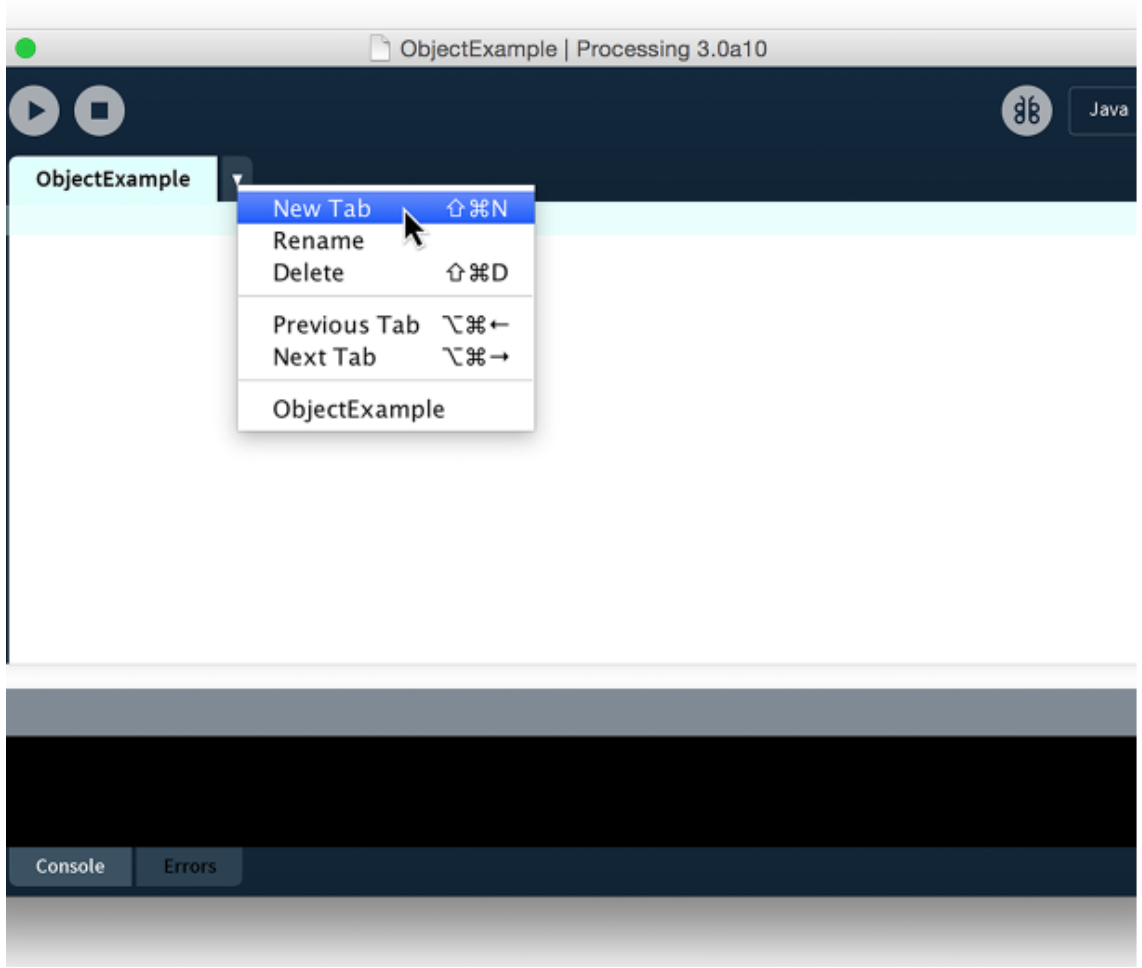
Functionality

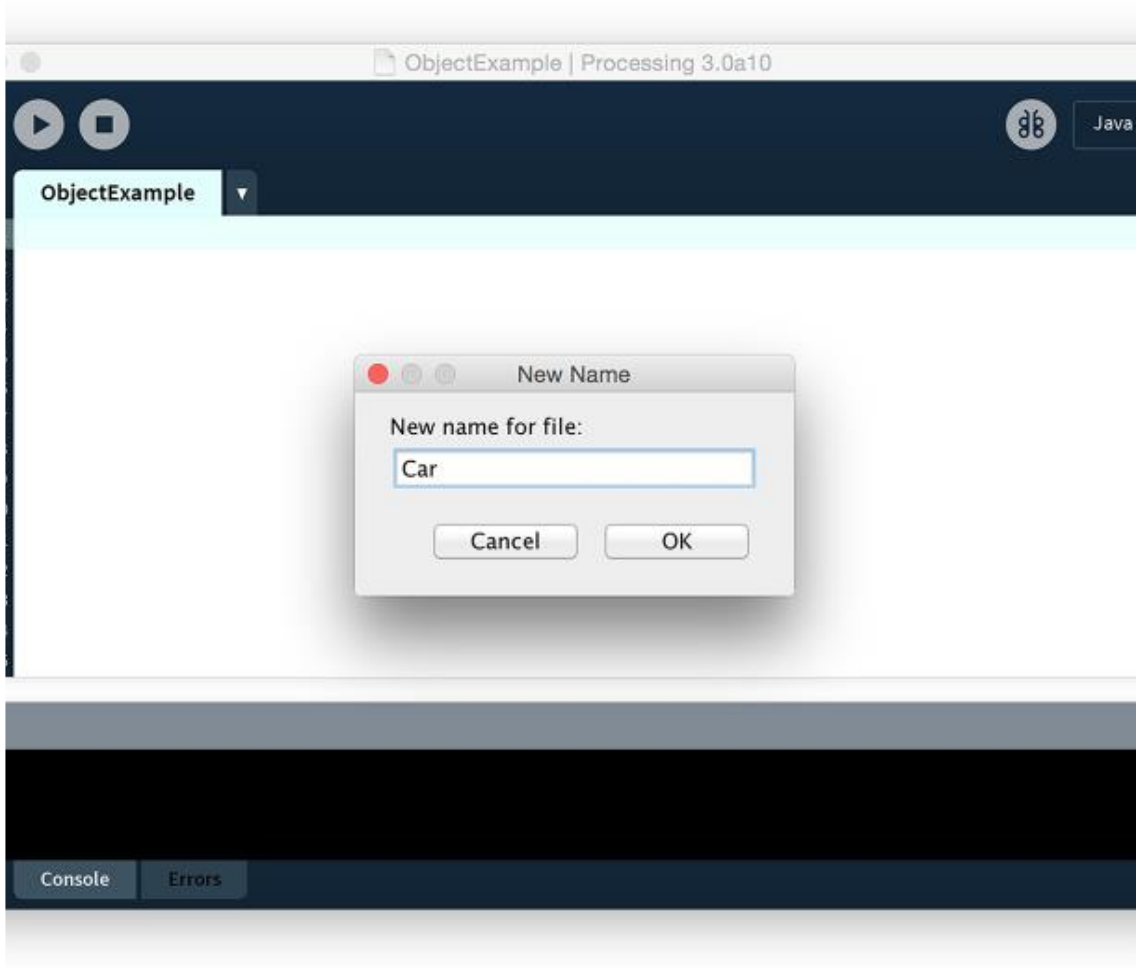
CREAZIONE DI UNA CLASSE

A class is a new block of code!

```
void setup() {

}

void draw() {

}

class Car {

}
```

CREAZIONE DI UNA CLASSE





AUTO CON CLASSI: DICHIARAZIONE

```
class Car{
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car(){
    c=color(255);
    xpos=width/2;
    ypos=height/2;
    xspeed=1;
  }

  void display(){
    rectMode(CENTER);
    fill(c);
    rect(xpos,ypos,20,10);
  }

  void drive(){
    xpos=xpos+xspeed;
    if(xpos > width) xpos = 0;
  }
}
```

USO DI UNA CLASSE

```
Car auto;

void setup(){
  size(400,400);
  auto = new Car();
}

void draw(){
  background(0);
  auto.drive();
  auto.display();
}
```

COSTRUTTORE

- Dall'esempio precedente posso creare diverse istanze della classe Car chiamando ripetutamente l'istruzione new
- Tutte le istanze create saranno identiche
- Per modificare le proprietà di ogni oggetto, possiamo passare degli argomenti al costruttore
  - Car myCar = new Car(color(255,0,0), 0, 100, 2);

PASSAGGIO DI PARAMETRI AL COSTRUTTORE

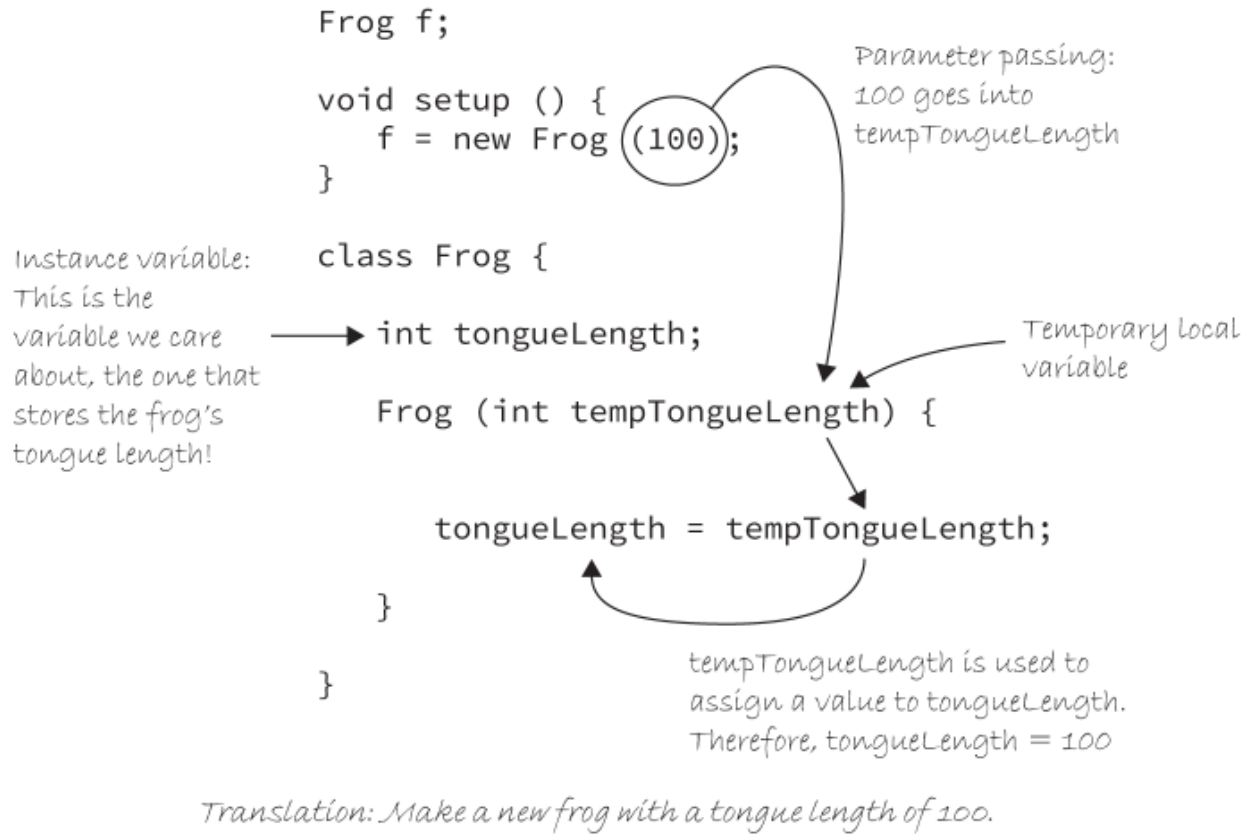


Figure 8-6

ARRAY

RIASSUNTO DEI PASSI NECESSARI PER UN ARRAY

- Dichiarazione
- Creazione
- Inizializzazione
- Indicizzazione/modifica/uso

ARRAY

- Un array rappresenta una lista di variabili che hanno un nome comune
- E' possibile utilizzare tante variabili senza creare un nome diverso per ognuna
  - Esempio: `ball1, ball2, ball3`
- Quando è necessario gestire tanti elementi dello stesso tipo, un array è una soluzione molto efficace e compatta
- Ogni array è composto da una lista di elementi; ogni elemento ha un indice che identifica la posizione dell'elemento nell'array

ARRAY

- Una variabile rappresenta un nome per una locazione in memoria
- Allo stesso modo, un array è un nome che rappresenta una lista di posizioni contigue in memoria
- Un array mantiene l'ordine degli elementi inseriti

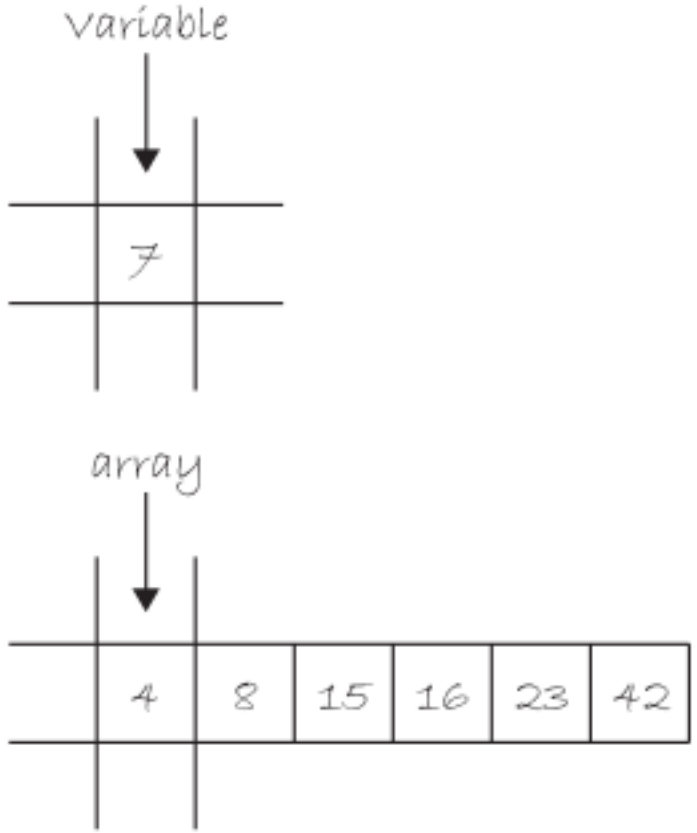


Figure 9-1

POSIZIONE DEGLI ELEMENTI

- Ogni elemento ha un indice all'interno della lista
- Il primo elemento è nella posizione 0 (zero)
  - Una distanza zero dall'inizio

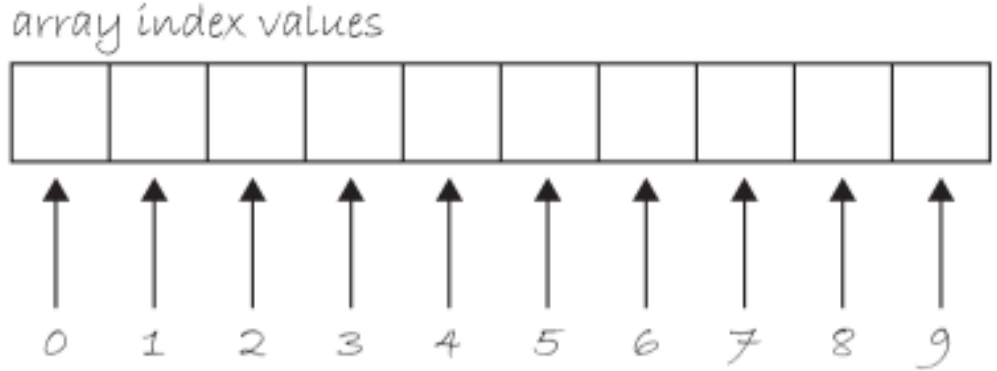


Figure 9-2

DICHIARAZIONE DI UN ARRAY

- La creazione di un array è simile alla dichiarazione di una nuova variabile: si usano le parentesi quadre per identificare il tipo della variabile
- Al momento della dichiarazione dell'array non è necessario specificare la lunghezza
- Una volta definita la lunghezza, questa non può più essere modificata

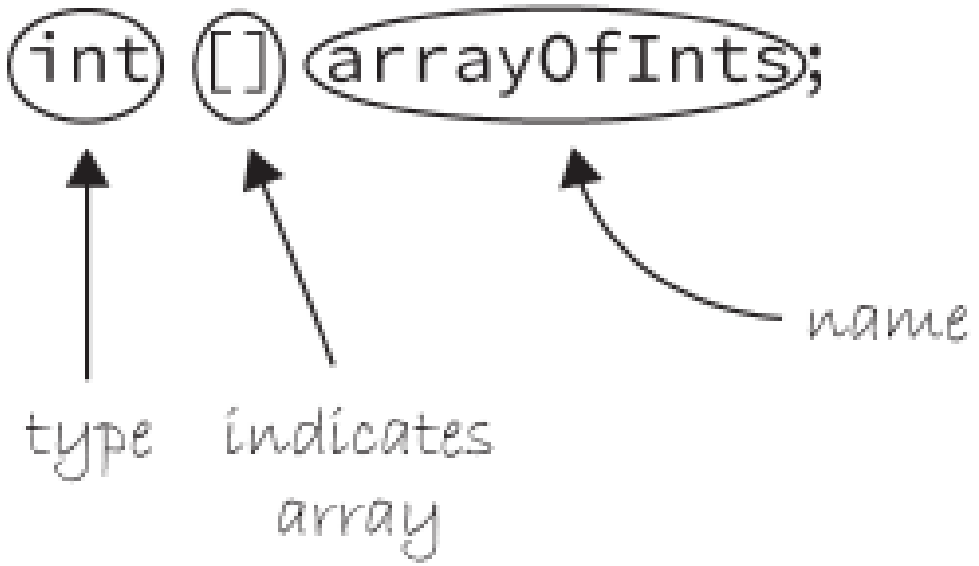


Figure 9-3

CREAZIONE DI UN ARRAY

- Per utilizzare un array appena dichiarato è necessario creare lo spazio necessario
- Bisogna decidere quante locazioni deve contenere
- Si usa di nuovo la parola chiave new come per gli oggetti

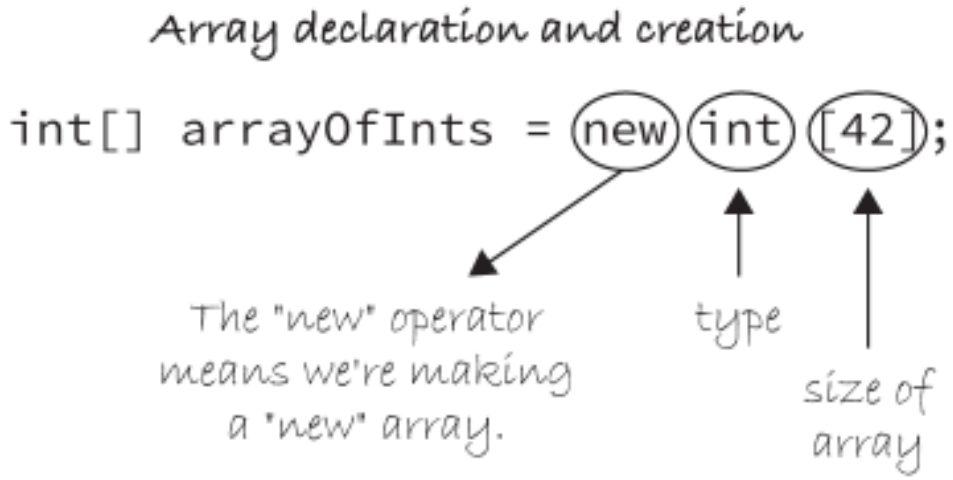


Figure 9-4

ESEMPI DI ARRAY

Example 9-1. Additional array declaration and creation examples

```
float[] scores = new float[4];                    // A list of 4 floating point numbers
Human[] people = new Human[100];                  // A list of 100 Human objects
int num = 50;
Car[] cars = new Car[num];                        // Using a variable to specify size
Spaceship[] ships = new Shapeship[num*2 + 3];     // Using an expression to
                                                      specify size
```

INIZIALIZZAZIONE DI UN ARRAY (1)

- Dopo la creazione dell'array abbiamo a disposizione un numero di locazioni fissato
- Il contenuto di ogni locazione è al momento non definito
- Bisogna inizializzare l'array con i valori iniziali che l'array dovrà contenere
- Abbiamo due modi. Inizializzazione di ogni singola posizione:

Example 9-2. Initializing the elements of an array one at a time

```
int[] stuff = new int[3];

stuff[0] = 8; // The first element of the array equals 8
stuff[1] = 3; // The second element of the array equals 3
stuff[2] = 1; // The third element of the array equals 1
```

INIZIALIZZAZIONE DI UN ARRAY

- Secondo metodo: lista di valori tra parentesi graffe

Example 9-3. Initializing the elements of an array all at once

```
int[] arrayOfInts = { 1, 5, 8, 9, 4, 5 } ;
float[] floatArray = { 1.2, 3.5, 2.0, 3.4123, 9.9 } ;
```

ARRAY E CICLI

- Posso sfruttare i cicli per scorrere l'array per modificare i valori, sfruttando una variabile contatore
- Utilizzando un ciclo `for`:

```
for (int i = 0; i < 1000; i++){
  values[i] = random(0,10);
}
```

- Oppure un ciclo `while`:

```
int n = 0;
while(n < 1000){
  values[n] = random(0,10);
  n = n + 1;
}
```

- La proprietà `length` può essere usata anche in Processing

ESERCIZIO: CREAZIONE DI UNA SCIA DEL MOUSE

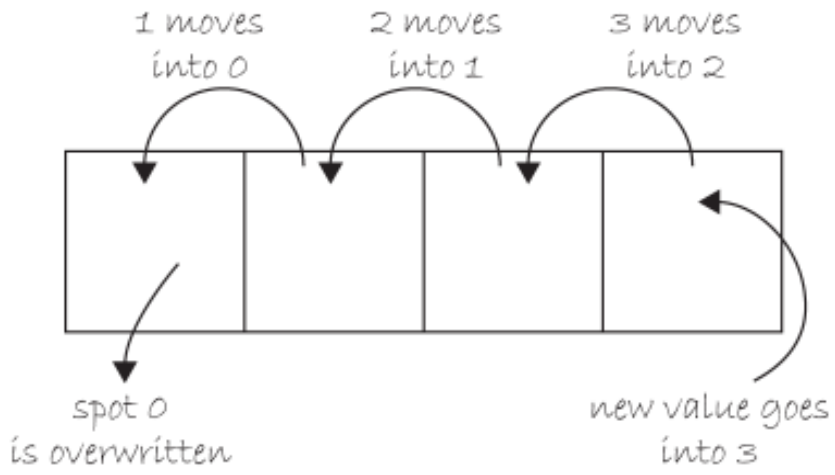
- Si vuole realizzare una striscia che si aggiorna al passaggio del mouse per simulare una scia del suo movimento
- E' necessario mantenere uno storico delle ultime posizioni per ricostruire una linea dalla posizione attuale a quella più vecchia
- Le posizione si gestiscono con due array, uno per le posizioni x e una per le posizioni y, entrambi inizializzati a zero

```
int[] xpos = new int[50];
int[] ypos = new int[50];
```

```
for (int i = 0; i < xpos.length; i++) {
  xpos[i] = 0;
  ypos[i] = 0;
}
```

ESERCIZIO: CREAZIONE DI UNA SCIA DEL MOUSE

- Ad ogni ciclo del metodo draw() bisogna aggiornare i due array delle posizioni salvando la posizione del mouse (mouseX e mouseY)
- La posizione corrente viene memorizzata nella ultima posizione dell'array
- Prima di scrivere i nuovi valori vanno spostati i valori precedenti verso sinistra (shift)



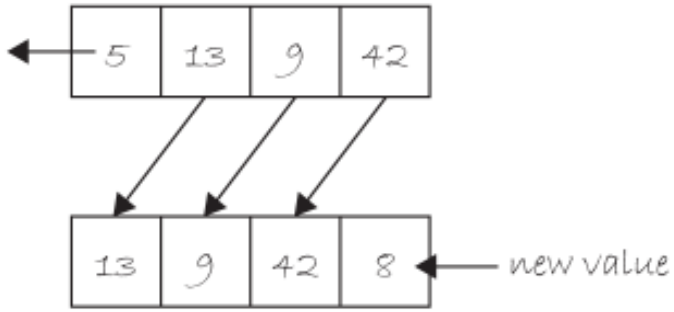


Figure 9-5

ESERCIZIO 9-8: CREAZIONE DI UNA SCIA DEL MOUSE

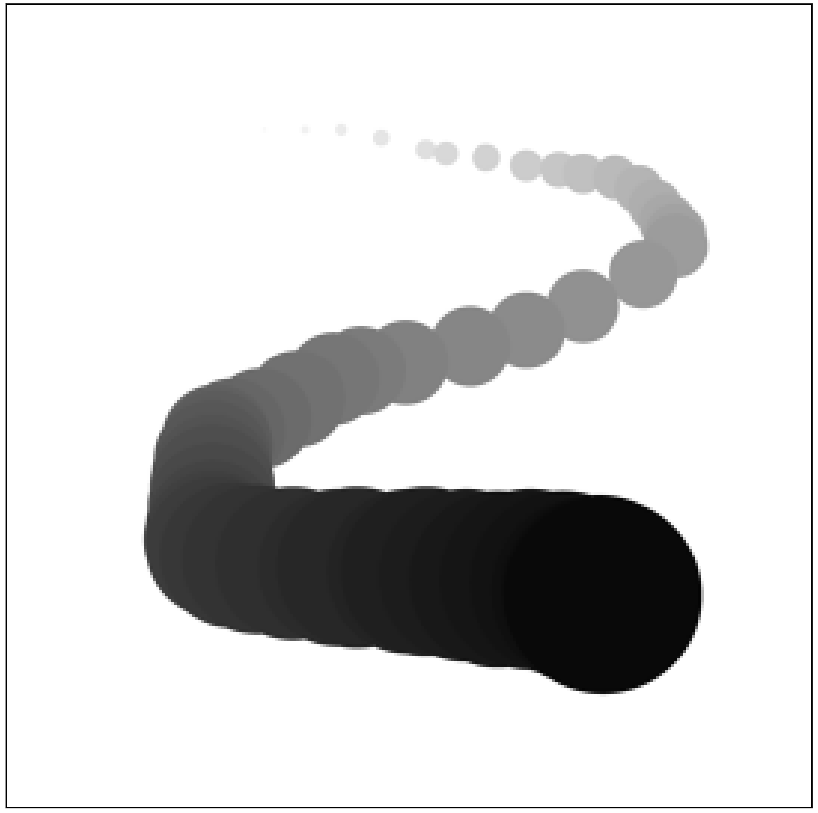


Figure 9-6

ARRAY DI OGGETTI

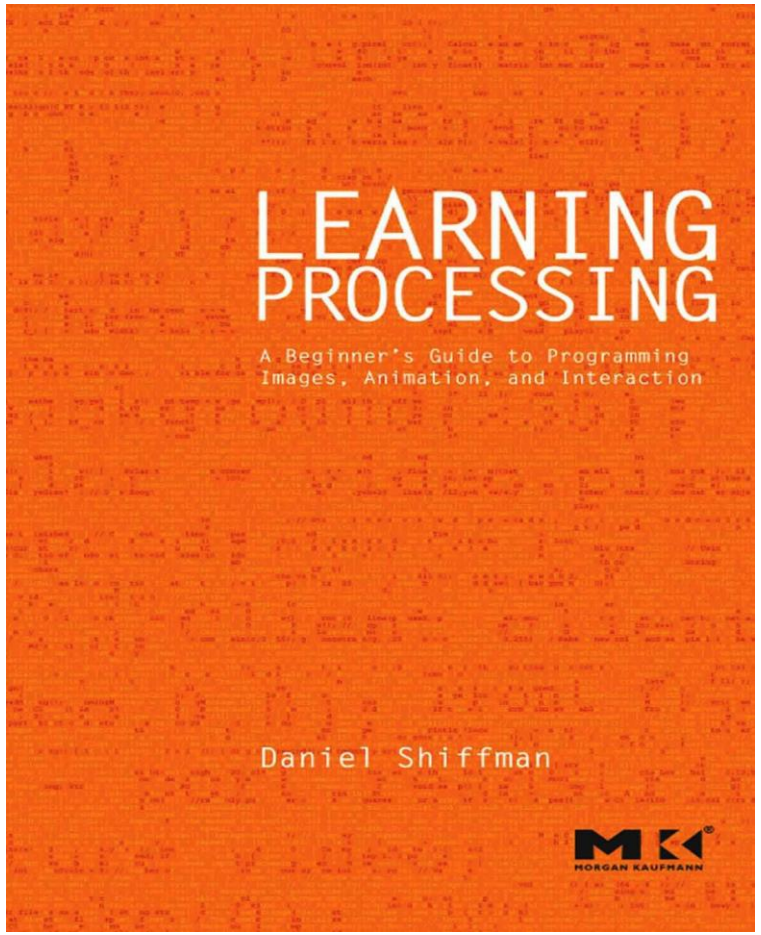
Before	After
`// Declare the car` `Car myCar;`	`// Declare the car array` `Car[] cars = new Car[100];`

`// Initialize the car`
`myCar = new Car(color(255), 0, 100, 2);`
	`// Initialize each element of the array`
	`for (int i = 0; i < cars.length; i++) {`
	`  cars[i] = new Car(color(i*2), 0, i*2, i);`
	`}`

`// Run the car by calling methods`
`myCar.move();`
`myCar.display();`
	`// Run each element of the array`
	`for (int i = 0; i < cars.length; i++) {`
	`  cars[i].move();`
	`  cars[i].display();`
	`}`

IMMAGINI

LIBRI E RIFERIMENTI



- Capitolo 15

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

IMMAGINI

- Processing permette la visualizzazione di immagini attraverso la classe predefinita Pimage
- Per visualizzare una immagine sulla finestra di disegno è necessario preparare il file da visualizzare
- Per caricare una immagine, bisogna importarla all'interno dello sketch, dal menu `Sketch -> Add file…`

CARICAMENTO DI UNA IMMAGINE NELLO SKETCH

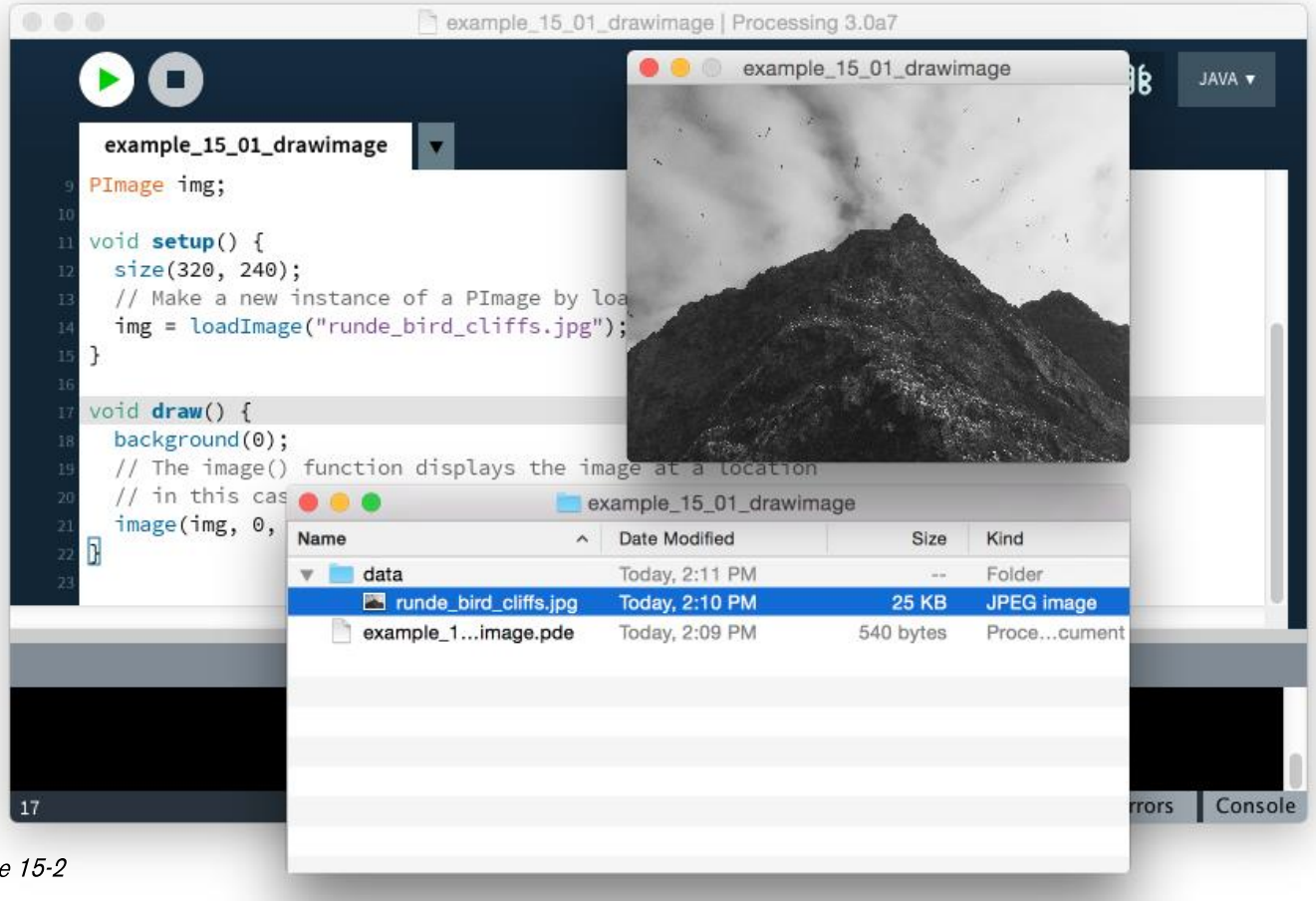


Figure 15-2

ESEMPIO DI VISUALIZZAZIONE

Example 15-1. "Hello World" images

```
PImage img;

void setup() {
  size(320, 240);
  img = loadImage("runde_bird_cliffs.jpg");
}

void draw() {
  background(0);
  image(img, 0, 0);
}
```

Declare a variable of type `PImage`, a class available from the Processing core library.

Make a new instance of a `PImage` by loading an image file.

The `image()` function displays the image at a location—in this case the point (0,0).

SPRITE – ANIMAZIONI CON LE IMMAGINI

Example 15-2. Image "sprite"

```
PImage head; // A variable for the image file
float x, y;  // Variables for image location
float rot;   // A variable for image rotation

void setup() {
  size(200, 200);
  // Load image, initialize variables
  head = loadImage("face.jpg");
  x = 0;
  y = width/2;
  rot = 0;
}

void draw() {
  background(255);

  translate(x, y);
  rotate(rot);
  imageMode(CENTER);
  image(head, 0, 0);

  // Adjust variables for animation
  x += 1.0;
  rot += 0.01;
  if (x > width) {
    x = 0;
  }
}
```

Images can be animated just like regular shapes using variables, `translate()`, `rotate()`, and so on.

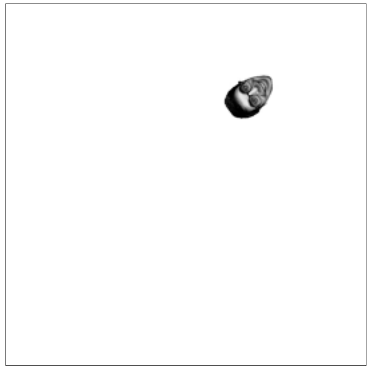


Figure 15-3

FILTRI

Figure 15-4

```
tint(255);
image(sunflower, 0, 0);
```

A The image retains its original state.

```
tint(100);
image(sunflower, 0, 0);
```

B The image appears darker.

```
tint(255, 127);
image(sunflower, 0, 0);
```

C The image is at 50% opacity.

```
tint(0, 200, 255)
image(sunflower, 0, 0);
```

D None of it is red, most of it is green, and all of it is blue.

```
tint(255, 0, 0, 100);
image(sunflower, 0, 0);
```

E The image is tinted red and transparent.

ARRAY DI IMMAGINI

```
// Image array
PImage[] images = new PImage[5];
```

```
// Loading images into an array
images[0] = loadImage("cat.jpg");
images[1] = loadImage("mouse.jpg");
images[2] = loadImage("dog.jpg");
images[3] = loadImage("kangaroo.jpg");
images[4] = loadImage("porcupine.jpg");
```

```
// Loading images into an array from an array of filenames
String[] filenames = {"cat.jpg", "mouse.jpg", "dog.jpg", "kangaroo.jpg",
"porcupine.jpg");
for (int i = 0; i < filenames.length; i++) {
  images[i] = loadImage(filenames[i]);
}
```

```
// Loading images with numbered files
for (int i = 0; i < images.length; i++) {
  images[i] = loadImage("animal" + i + ".jpg");
}
```

PIXELS

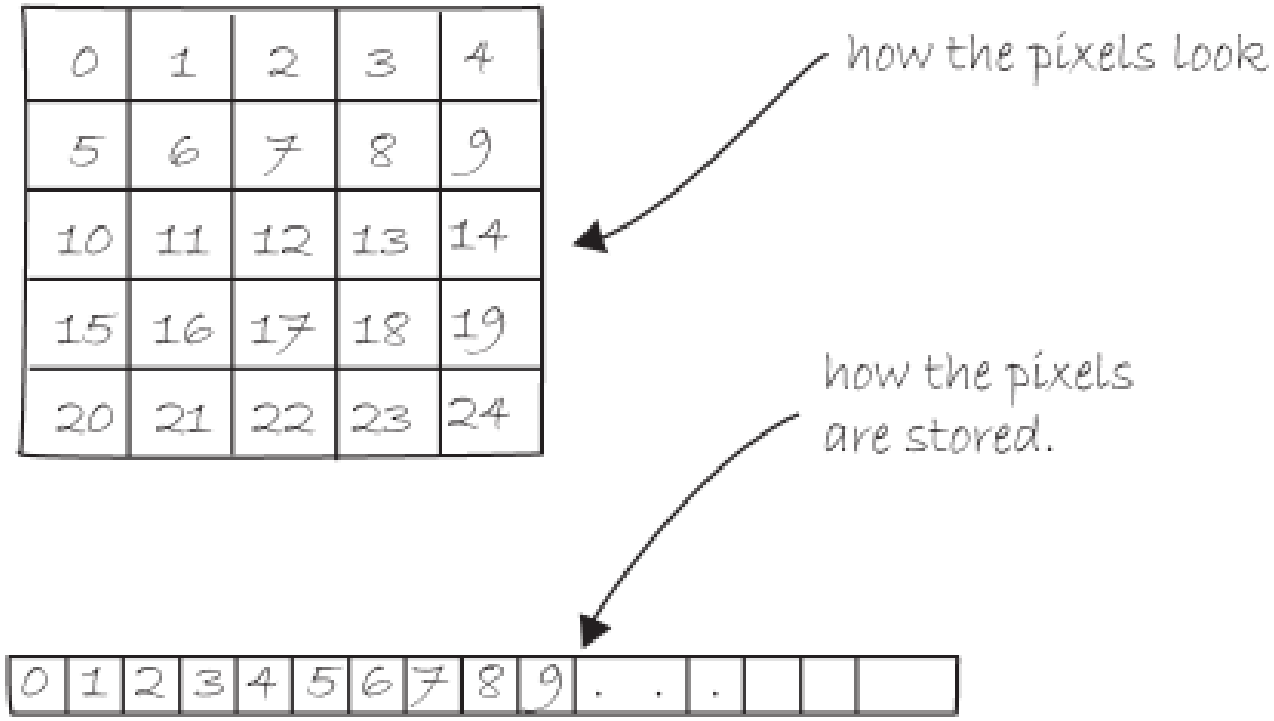


Figure 15-5

ESEMPIO 15-5

Example 15-5. Setting pixels

```
size(200, 200);
// Before I deal with pixels
loadPixels();

// Loop through every pixel
for (int i = 0; i < pixels.length; i++) {
```

You can get the length of the pixels array just like with any array.

```
  // Pick a random number, 0 to 255
  float rand = random(255);
  // Create a grayscale color based on random number
  color c = color(rand);
  // Set pixel at that location to random color
  pixels[i] = c;
}
```

You can access individual elements of the pixels array via an index, just like with any other array.

```
// When you are finished dealing with pixels
updatePixels();
```

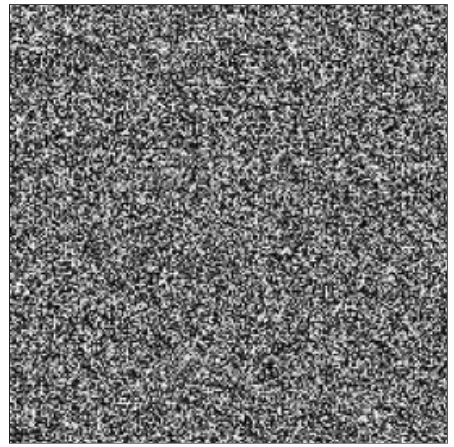


Figure 15-6

POSIZIONE DI OGNI PIXEL

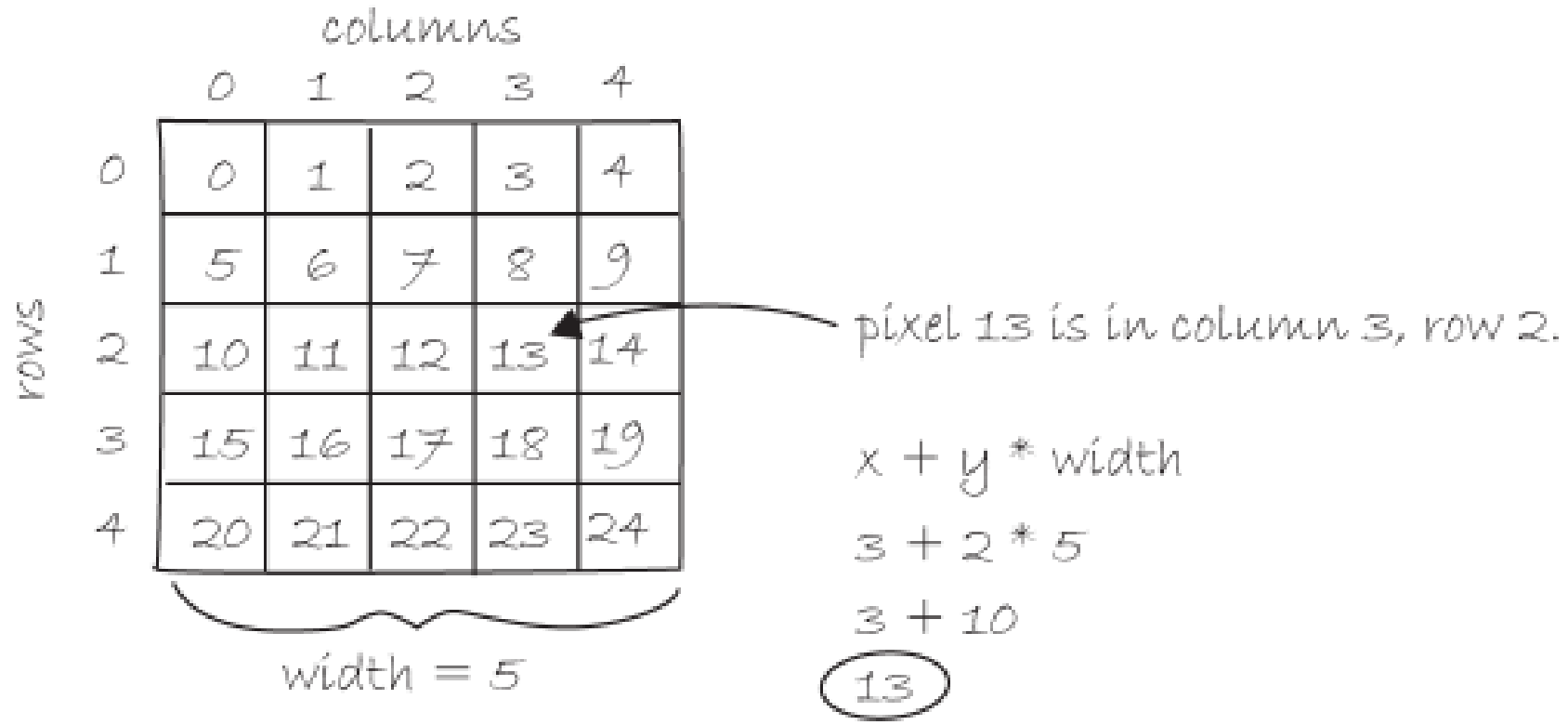


Figure 15-7

ESEMPIO 15-6 – GRIGLIA DI PIXEL

Example 15-6. Setting pixels according to their 2D location

```
size(200, 200);
loadPixels();

// Loop through every pixel column
for (int x = 0; x < width; x++) {

  // Loop through every pixel row
  for (int y = 0; y < height; y++) {

    int loc = x + y * width;



    if (x % 2 == 0) {
      pixels[loc] = color(255);
    } else {
      pixels[loc] = color(0);
    }
  }
}
updatePixels();
```

Two loops allow you to visit every column (x) and every row (y).

The location in the pixel array is calculated via the formula: 1D pixel location = x + y * width

Using the column number (x) to determine whether the color should be black or white

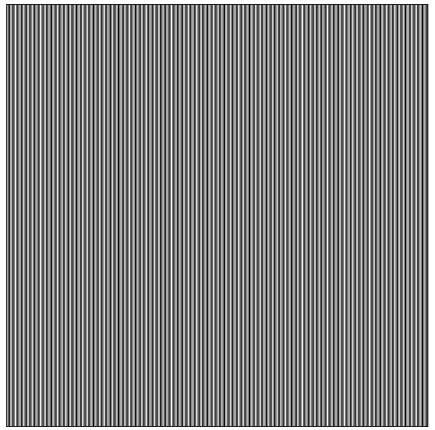


Figure 15-8

FILTRI PERSONALIZZATI

Example 15-7. Displaying the pixels of an image

```
PImage img;

void setup() {
  size(200, 200);
  img = loadImage("sunflower.jpg");
}

void draw() {
  loadPixels();
  img.loadPixels();

  for (int y = 0; y < height; y++) {
    for (int x = 0; x < width; x++) {
     int loc = x + y * width;
     float r = red  (img.pixels[loc]);
     float g = green(img.pixels[loc]);
     float b = blue (img.pixels[loc]);


     // image processing!
     // image processing!
     // image processing!


     // Set the display pixel
     pixels[loc] = color(r, g, b);
    }
  }
  updatePixels();
}
```

You must also call `loadPixels()` on the `PImage`.

The functions `red()`, `green()`, and `blue()` pull out the three color components from a pixel.

If the RGB values were to change, it would happen here, before setting the pixel in the display window.

Figure 15-9

MODIFICA DELLA LUMINOSITÀ DI UNA IMMAGINE

Example 15-8. Adjusting image brightness

```
for (int x = 0; x < img.width; x++) {
  for (int y = 0; y < img.height; y++) {
    // Calculate the 1D pixel location
    int loc = x + y * img.width;
    // Get the red, green, blue values
    float r = red  (img.pixels[loc]);
    float g = green(img.pixels[loc]);
    float b = blue (img.pixels[loc]);


    // Adjust brightness with mouseX
    float adjustBright
        = map(mouseX, 0, width, 0, 8);
    r *= adjustBright;
    g *= adjustBright;
    b *= adjustBright;

    r = constrain(r, 0, 255);
    g = constrain(g, 0, 255);
    b = constrain(b, 0, 255);

    // Make a new color
    color c = color(r, g, b);
    pixels[loc] = c;
  }
}
```

Figure 15-10

I calculate a multiplier ranging from 0.0 to 8.0 based on `mouseX` position using `map()`. That multiplier changes the RGB value of each pixel.

The RGB values are constrained between 0 and 255 before being set as a new color.

LUMINOSITÀ DI UNA IMMAGINE

Example 15-9. Adjusting image brightness based on pixel location

```
for (int x = 0; x < img.width; x++) {
  for (int y = 0; y < img.height; y++) {
    // Calculate the 1D pixel location
    int loc = x + y * img.width;
    // Get the red, green, blue values from pixel
    float r = red  (img.pixels[loc]);
    float g = green(img.pixels[loc]);
    float b = blue (img.pixels[loc]);

    // Calculate an amount to change brightness
    // based on proximity to the mouse
    float distance = dist(x, y, mouseX, mouseY);
    float adjustBright = map(distance, 0, 50, 8, 0);
    r *= adjustBrightness;
    g *= adjustBrightness;
    b *= adjustBrightness;
    // Constrain RGB to between 0-255
    r = constrain(r, 0, 255);
    g = constrain(g, 0, 255);
    b = constrain(b, 0, 255);
    // Make a new color
    color c = color(r, g, b);
    pixels[loc] = c;
  }
}
```

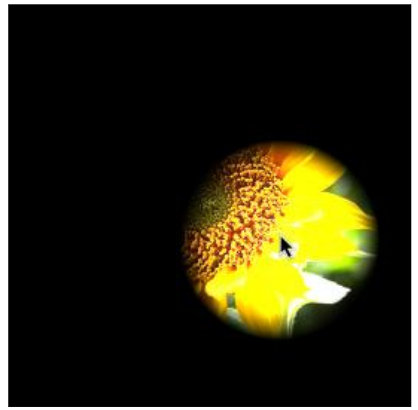


Figure 15-11

The closer the pixel is to the mouse, the lower the value of `distance` is. I want closer pixels to be brighter, however, so I invert the `adjustBrightness` factor using `map()`. Pixels with a distance of 50 (or greater) have their brightness multiplied by 0.0 (resulting in now brightness) and brightness for pixels with a distance of 0 is multiplied by a factor of 8.

CREAZIONE DI UNA NUOVA IMMAGINE

Example 15-10. Brightness threshold

```
PImage source;      // Source image
PImage destination; // Destination image

void setup() {
  size(200, 200);
  source = loadImage("sunflower.jpg");
  destination = createImage(source.width,
                            source.height, RGB);
}

void draw() {
  float threshold = 127;

  // The sketch is going to look at both image's pixels
  source.loadPixels();
  destination.loadPixels();
```

Two images are required, a source (original file) and destination (to be displayed) image.

The destination image is created as a blank image the same size as the source.

Figure 15-12

FUNZIONE `filter()`

Example 15-11. Brightness threshold with filter

```
// Draw the image
image(img, 0, 0);
// Filter the window with a threshold effect
// 0.5 means threshold is 50% brightness
filter(THRESHOLD, 0.5);
```

- La funzione filter() offre uan serie di filtri predefiniti:
  - GRAY, INVERT, POSTERIZE, BLUR, OPAQUE, ERODE, DILATE

MODIFICA DI PIXEL IN GRUPPO

Example 15-12. Pixel neighbor differences (edges)

```
// Since I am looking at left neighbors
// I skip the first column
for (int x = 1; x < width; x++) {
  for (int y = 0; y < height; y++) {
    // Pixel location and color
    int loc = x + y * img.width;
    color pix = img.pixels[loc];


    // Pixel to the left location and color
    int leftLoc = (x - 1) + y * img.width;
    color leftPix = img.pixels[leftLoc];
    // New color is difference between
    // pixel and left neighbor
    float diff = abs(brightness(pix)
        - brightness(leftPix));
    pixels[loc] = color(diff);
  }
}
```

Reading the pixel to the left.

Figure 15-13

FILTRI CONVOLUZIONE SPAZIALE

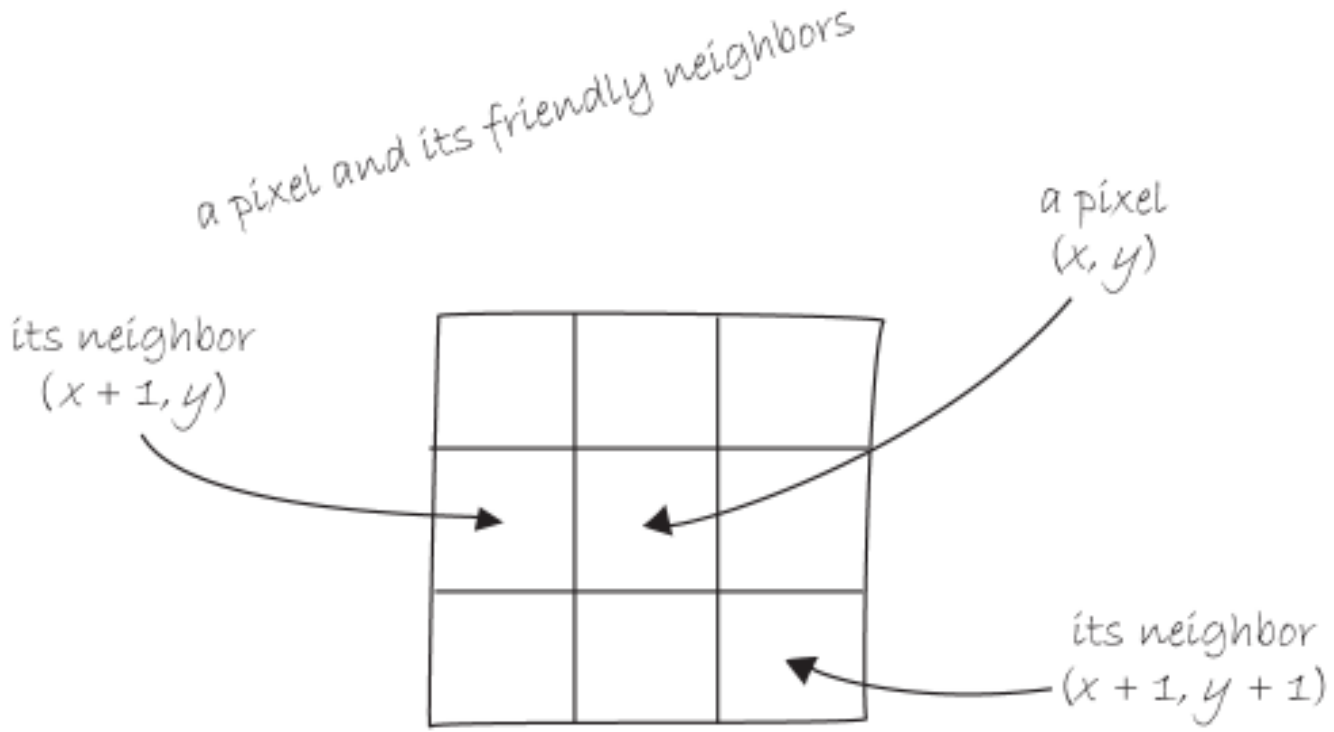


Figure 15-14

SHARPEN CON CONVOLUZIONE

Example 15-13. Sharpen with convolution

```
PImage img;
int w = 80;

// it's possible to perform a convolution
// the image with different matrices

float[][] matrix = { { -1, -1, -1 },
                     { -1,  9, -1 },
                     { -1, -1, -1 } } ;
void setup() {
  size(200, 200);
  img = loadImage("sunflower.jpg");
}

void draw() {
  // The sketch is only going to process a portion of the image
  // so let's set the whole image as the background first
  image(img, 0, 0);

  int xstart = constrain(mouseX - w/2, 0, img.width);
  int ystart = constrain(mouseY - w/2, 0, img.height);
  int xend   = constrain(mouseX + w/2, 0, img.width);
  int yend   = constrain(mouseY + w/2, 0, img.height);
  int matrixsize = 3;
```

The convolution matrix for a "sharpen" effect stored as a 3 × 3 two-dimensional array.

Figure 15-15

In this example only a section of the image—an 80 × 80 rectangle around the mouse location—is processed.

COMPOSIZIONI CREATIVE

Example 15-14. "Pointillism"

```
PImage img;
int pointillize = 16;

void setup() {
  size(200, 200);
  img = loadImage("sunflower.jpg");
  background(0);
}

void draw() {
  // Pick a random point
  int x = int(random(img.width));
  int y = int(random(img.height));
  int loc = x + y * img.width;

  // Look up the RGB color in the source image
  img.loadPixels();
  float r = red(img.pixels[loc]);
  float g = green(img.pixels[loc]);
  float b = blue(img.pixels[loc]);

  noStroke();
  fill(r, g, b, 100);
  ellipse(x, y, pointillize, pointillize);
}
```

Back to shapes! Instead of setting a pixel, use the color from a pixel to draw a circle.

Figure 15-16