

FONDAMENTI DI INFORMATICA

Alma Artis

Francesca Pratesi (ISTI, CNR)

Processing – Variabili, interazioni ed eventi di base

RICAPITOLANDO

- `background(color);`
- `stroke(color);`
- `fill(color);`
- `noFill();`
- `point(x,y);`
- `line(start_x, start_y, end_x, end_y)`
- `// se rectMode(CENTER)`
- `rect(center_x, center_y, width, height);`
- `// se rectMode(CORNER)`
- `rect(top_left_x, top_left_y, width, height);`
- `// + Ellipse`



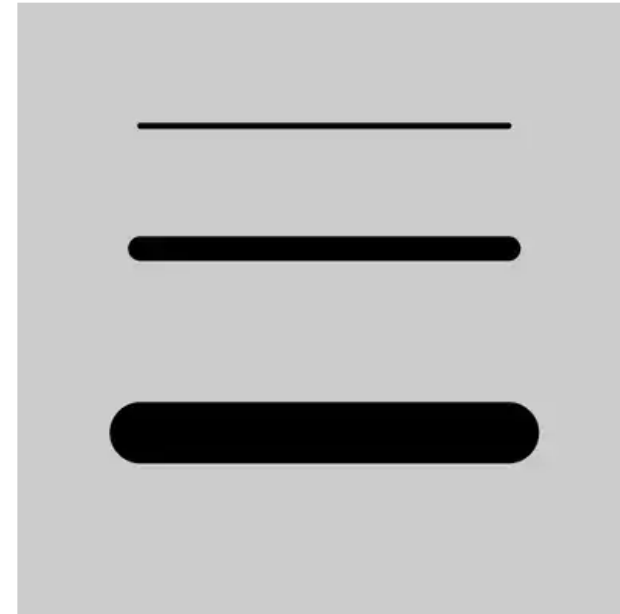
ERRATA CORRIGE

- La scorsa lezione vi avevo detto che se `rectMode(CORNER)`
- `rect(top_left_x, top_left_y, bottom_right_x, bottom_right_y);`
- In realtà, queste impostazioni non sono più vere, ma dobbiamo dare la larghezza e l'altezza anche in CORNER-mode:
- `rect(top_left_x, top_left_y, width, height);`
- Questo è vero anche per le ellissi



SPESSORE DELLE LINEE

```
size(400, 400);  
strokeWeight(4); // Default  
line(80, 80, 320, 80);  
strokeWeight(16); // Thicker  
line(80, 160, 320, 160);  
strokeWeight(40); // Beastly  
line(80, 280, 320, 280);
```



COMMENTI

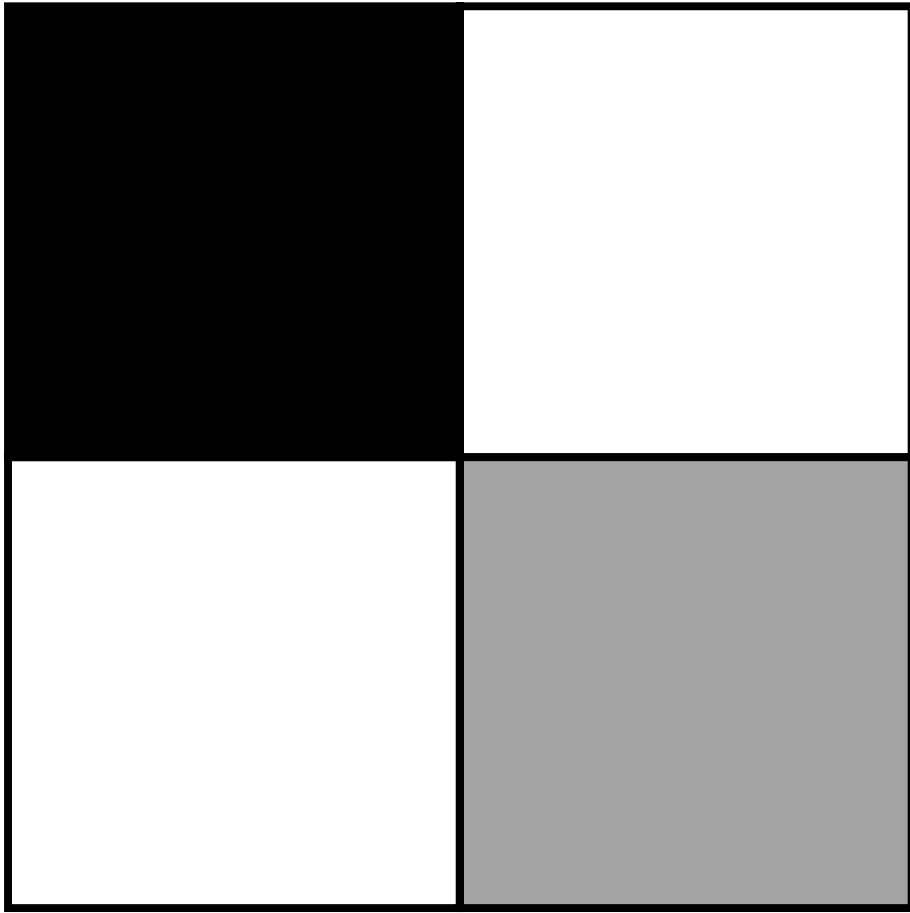
- In Processing, i commenti si scrivono come in Javascript

// questo è un commento su una sola riga

/* questo è un commento
su più righe */

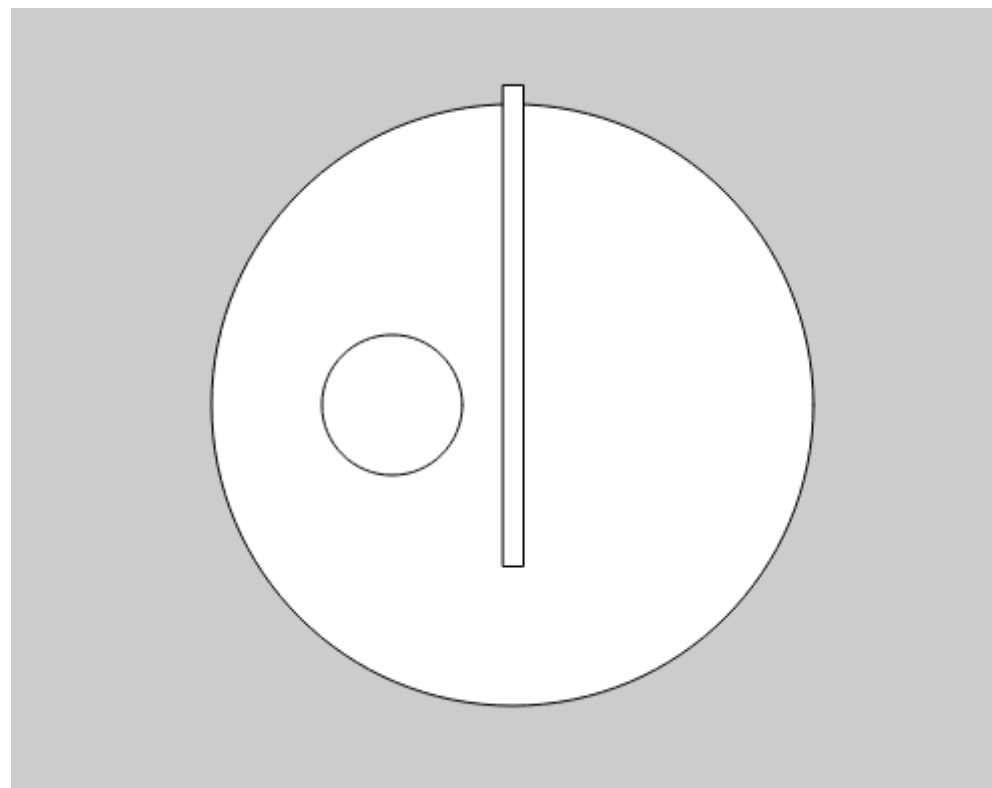
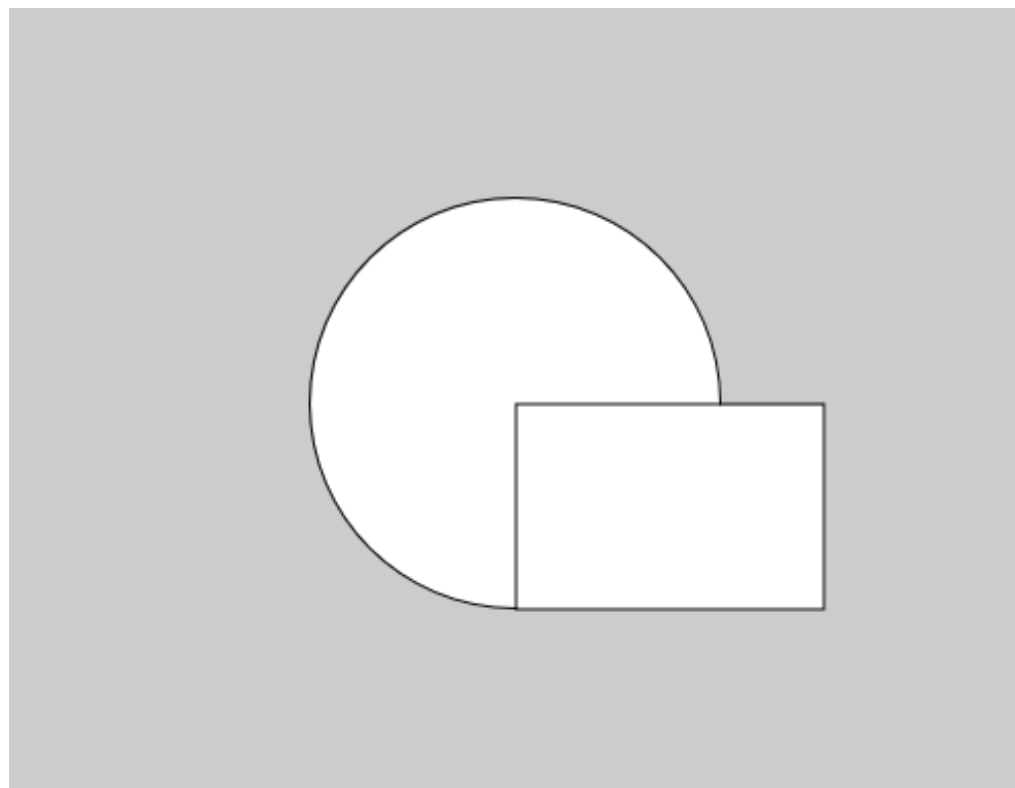


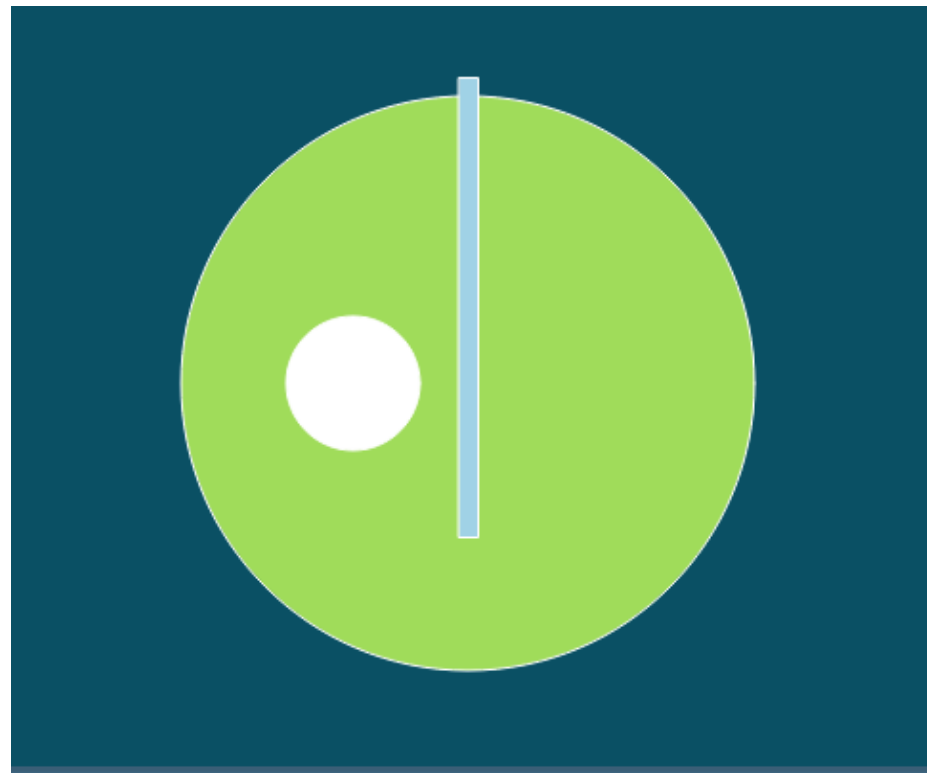
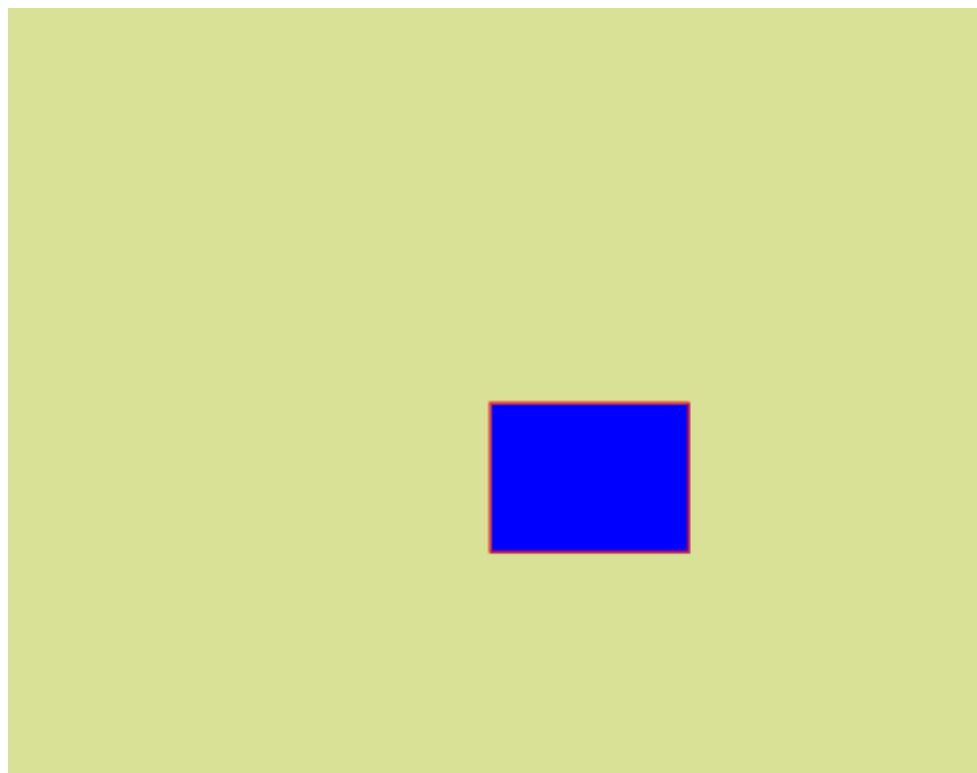
ESERCIZIO



- Scrivere le istruzioni per creare il diagramma a sinistra









INTERACTION

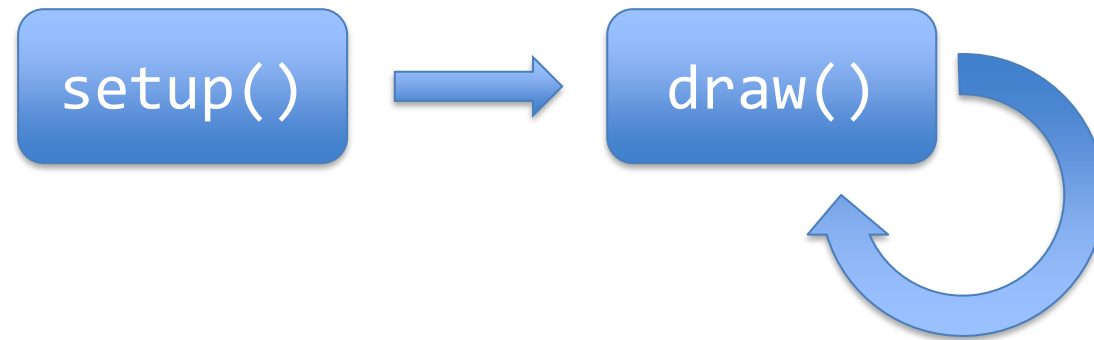
FLUSSO DI UN PROGRAMMA PROCESSING

- Due fasi principali
 - SETUP
 - Questa parte viene eseguita all'inizio e ha lo scopo di configurare lo sketch per l'esecuzione
 - DRAW
 - Questa parte disegna la finestra ciclicamente, più volte al secondo. Aggiornando la visualizzazione si aggiorna anche lo stato del sistema
- Esempio: considera la corsa di una maratona
 - SETUP: indossa le scarpe e l'abbigliamento sportivo
 - DRAW: metti un piede davanti all'altro. Dopo 42km fermati.



FUNZIONI `setup()` E `draw()`

- `setup()` prepara lo spazio di visualizzazione e inizializza lo stato interno del programma. Viene eseguita solo una volta!
- `draw()` viene chiamata ripetutamente fino a quando il programma è in esecuzione. Ha il compito di aggiornare lo stato interno del programma e la visualizzazione
 - Sfruttiamo le continue esecuzioni di `draw()` per monitorare lo stato del sistema



BLOCCO DI CODICE

- Un **blocco** è un insieme di istruzioni racchiuse tra parentesi graffe {}
- Un blocco può contenere al suo interno altri blocchi
 - Di solito i blocchi annidati dentro altri sono pure indentati, ovvero hanno una rientranza nell'inizio della linea di codice
- Due blocchi di codice rilevanti sono **setup()** e **draw()**



FUNZIONI `setup()` E `draw()`

What's this?

What are these for?

```
void setup() {  
    // Initialization code goes here  
}
```

void draw() {
 // Code that runs forever goes here
}

Curly brackets open and close a block of code.

fig. 3.1

FUNZIONI `setup()` E `draw()`

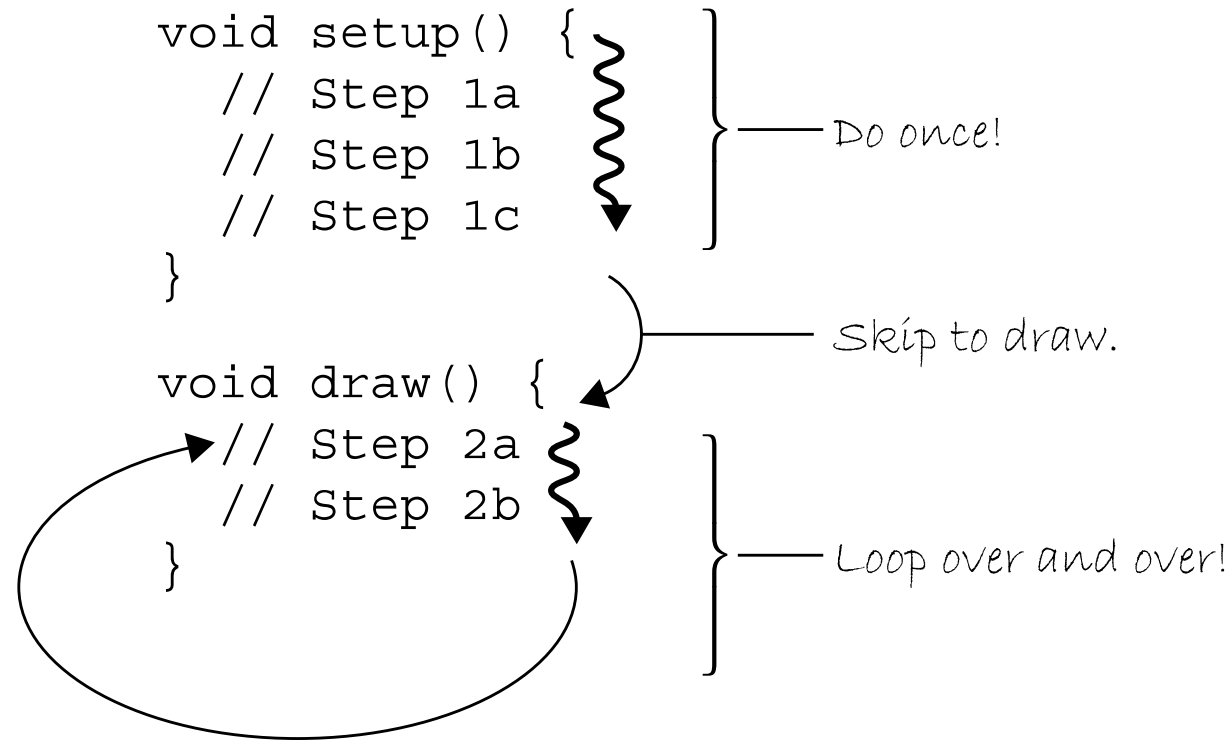


fig. 3.2

ESERCIZIO: ZOOG

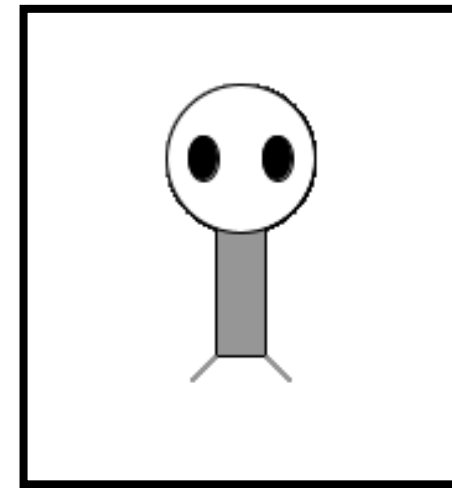
```
void setup(){  
  size(200,300);  
  background(255);  
}
```

setup() viene eseguita solo la prima volta.

size() dovrebbe essere sempre la prima linea di setup → Processing non può fare nulla finché la dimensione della finestra non è stata specificata

```
void draw(){  
  
  ellipseMode(CENTER);  
  rectMode(CENTER);  
  
  stroke(0); // resetto il colore delle linee  
  fill(150);  
  rect(100,100,20,100); // corpo  
  
  fill(255);  
  ellipse(100,70,60,60); // testa  
  
  fill(0);  
  ellipse(85,70,12,18); // occhio sx  
  ellipse(115,70,12,18); // occhio dx  
  
  stroke(150);  
  line(90,150,80,160); // gamba sx  
  line(110,150,120,160); // gamba dx  
}
```

draw() viene eseguita continuamente, finché non si chiude la finestra; in pratica, ridisegniamo la figura ogni volta sopra se stessa



VARIAZIONI CON IL MOUSE

- Le variabili `mouseX` e `mouseY` sono due variabili di sistema.
- Sono aggiornate automaticamente per rappresentare la posizione orizzontale e verticale del puntatore del mouse sullo sketch
- Le variabili `pmouseX` e `pmouseY` sono aggiornate con la posizione del mouse alla iterazione precedente



VARIAZIONI CON IL MOUSE

Example 3-2: *mouseX* and *mouseY*

```
void setup() {  
  size(200,200);  
}  
  
void draw() {  
  background(255);  
  
  // Body  
  stroke(0);  
  fill(175);  
  rectMode(CENTER);  
  rect(mouseX,mouseY,50,50);  
}
```

Try moving ***background()*** to ***setup()*** and see the difference! (Exercise 3–3)

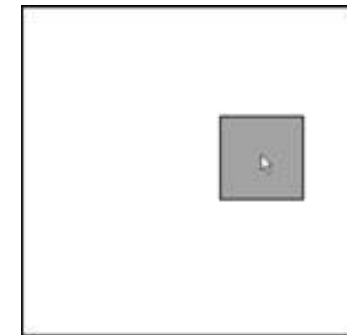


fig. 3.4

mouseX is a keyword that the sketch replaces with the horizontal position of the mouse.
mouseY is a keyword that the sketch replaces with the vertical position of the mouse.

Cosa accade se sposto l'istruzione **background(255)** in **setup()** ?

ESERCIZIO: TRACCIA DEL MOUSE

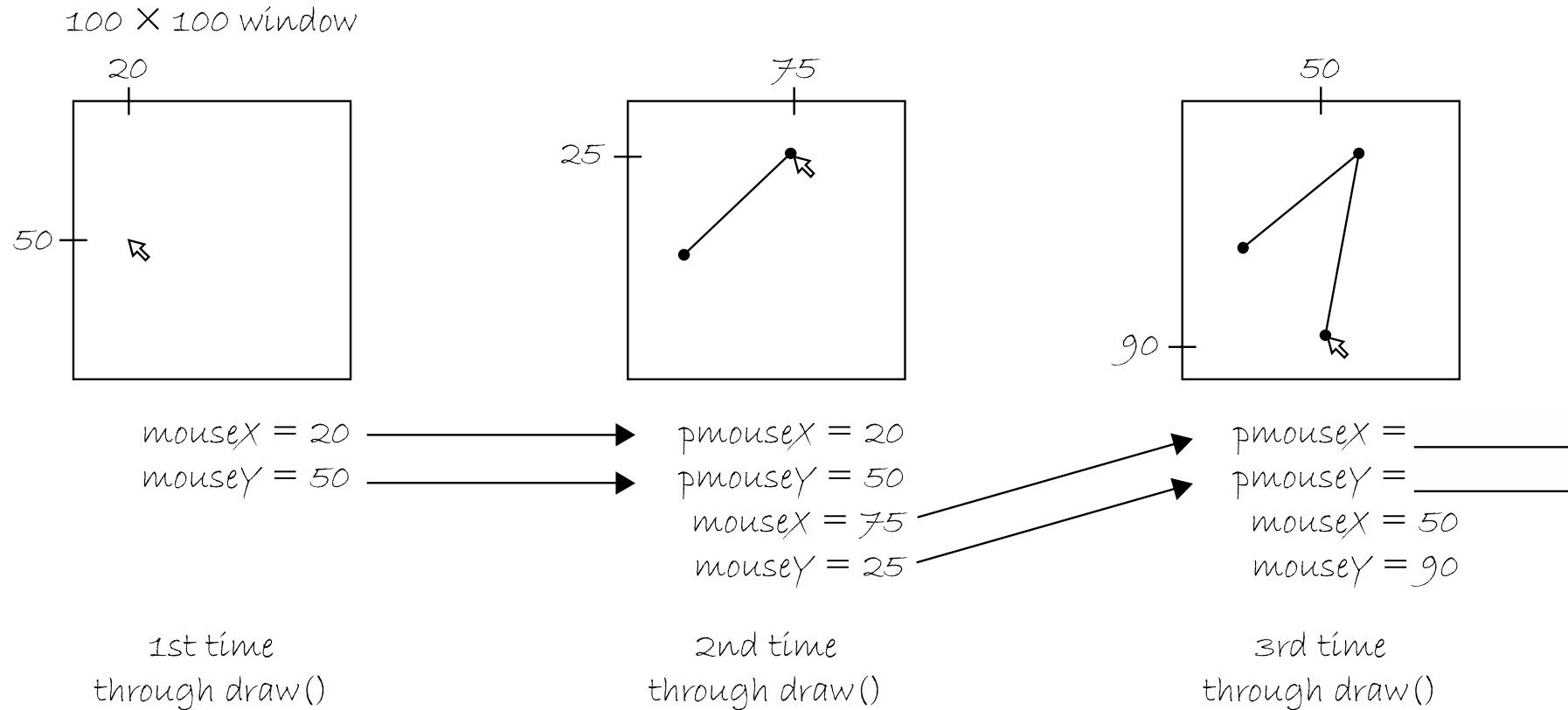


fig. 3.6

ESERCIZIO: DISEGNO DI UNA LINEA CONTINUA

Example 3-4: Drawing a continuous line

```
void setup() {  
  size(200,200);  
  background(255);  
  smooth();  
}  
  
void draw() {  
  stroke(0);  
  line(pmouseX,pmouseY,mouseX,mouseY);  
}
```

Draw a line from previous mouse location to current mouse location.

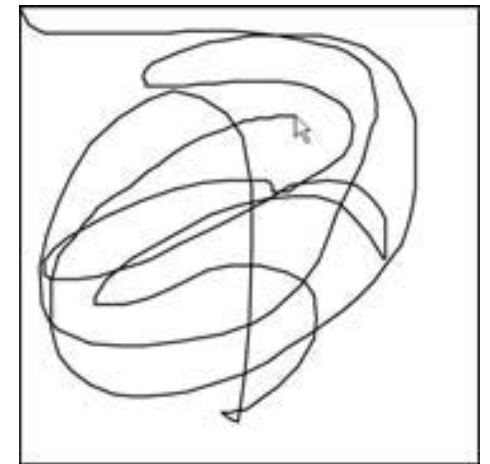


fig. 3.7

ESERCIZIO: LINEA CONTINUA CON TRATTO PROPORZIONALE ALLA VELOCITÀ



SOLUZIONE: DISEGNO DI UNA LINEA CON TRATTO PROPORZIONALE ALLA VELOCITÀ

- Partiamo dal codice per tracciare una linea continua

```
void setup(){  
    size(400,400);  
    background(255);  
    smooth();  
}  
  
void draw(){  
    stroke(0);  
    line(pmouseX,pmouseY,mouseX,mouseY);  
}
```



SOLUZIONE: DISEGNO DI UNA LINEA CON TRATTO PROPORZIONALE ALLA VELOCITÀ (2)

- Possiamo derivare la velocità del mouse dalla differenza tra la posizione corrente e quella precedente
- Diamo questo valore alla dimensione della linea

```
void setup(){  
    size(400,400);  
    background(255);  
    smooth();  
}  
  
void draw(){  
    stroke(0);  
    strokeWeight( (mouseX-pmouseX)+(mouseY-pmouseY)  
);  
    line(pmouseX,pmouseY,mouseX,mouseY);  
}
```



SOLUZIONE: DISEGNO DI UNA LINEA CON TRATTO PROPORZIONALE ALLA VELOCITÀ (3)

- Il cambiamento deve avvenire anche se ci spostiamo verso sinistra e verso l'alto
- Con la funzione `abs()` prendiamo il valore assoluto della differenza tra le posizioni

```
void setup(){  
    size(400,400);  
    background(255);  
    smooth();  
}  
  
void draw(){  
    stroke(0);  
    strokeWeight( (abs(mouseX-pmouseX)+abs(mouseY-  
pmouseY))/2 );  
    line(pmouseX,pmouseY,mouseX,mouseY);  
}
```



EVENTI: MOUSE E TASTIERA

- Il ciclo di esecuzione di `draw()` ci permette di rilevare la posizione del mouse tramite le variabili `mouseX` e `mouseY`
- Quando il bottone del mouse viene premuto viene generato un `evento`
- Quando si verifica un determinato evento, processing esegue una particolare funzione associata ad esso
- In risposta al click del mouse, viene chiamata la funzione `mousePressed()`
- In risposta alla pressione del tasto della tastiera viene chiamata la funzione `keyPressed()`



ESERCIZIO: DISEGNO DI PUNTI AL CLICK DEL MOUSE

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

```
void draw(){  
}
```

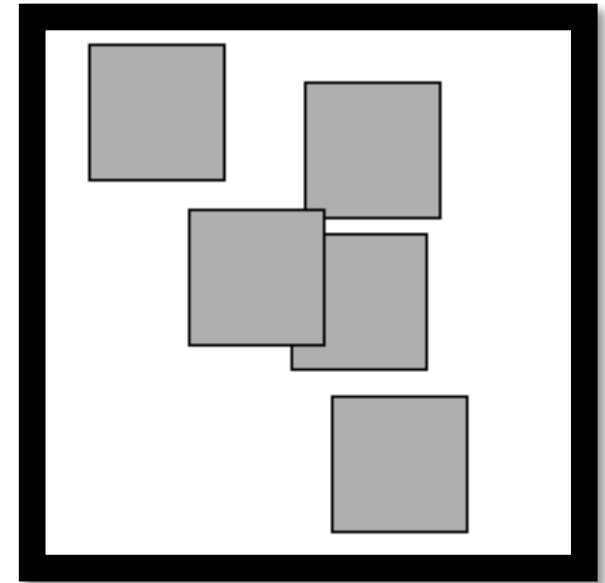
In questo esempio draw() è vuota: non vogliamo che venga fatto nulla in automatico

```
void mousePressed(){  
  stroke(0);  
  fill(175);  
  rectMode(CENTER);  
  rect(mouseX,mouseY,50,50);  
}
```

mousePressed() viene eseguita quando l'utente preme un pulsante del mouse

```
void keyPressed(){  
  background(255);  
}
```

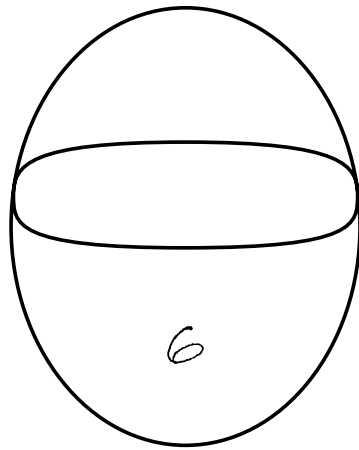
keyPressed() viene eseguita quando l'utente preme un pulsante della tastiera



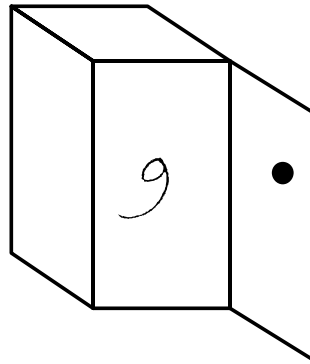


VARIABILI

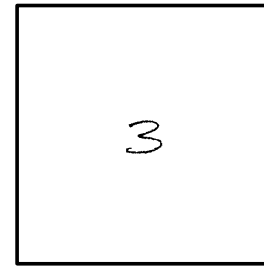
COSA È UNA VARIABILE?



variable
bucket



variable
locker



variable
post-it

fig. 4.1

Un nome per indirizzare una locazione della memoria del computer

USO DI VARIABILI

Jane's Score	Billy's Score
5	10
30	25
53	47
65	68
87	91
101	98

fig. 4.3

- Oltre a memorizzare un valore, una variabile permette di seguire il cambiamento del suo contenuto

DICHIARAZIONE DI VARIABILI

- Una variabile può memorizzare valori primitivi o riferimenti ad array e oggetti
- Una variabile deve essere dichiarata prima di essere usata
- Per dichiararla, bisogna specificare
 - il tipo: quale dato dovrà contenere
 - Il nome: come chiamiamo la locazione che conserva il dato

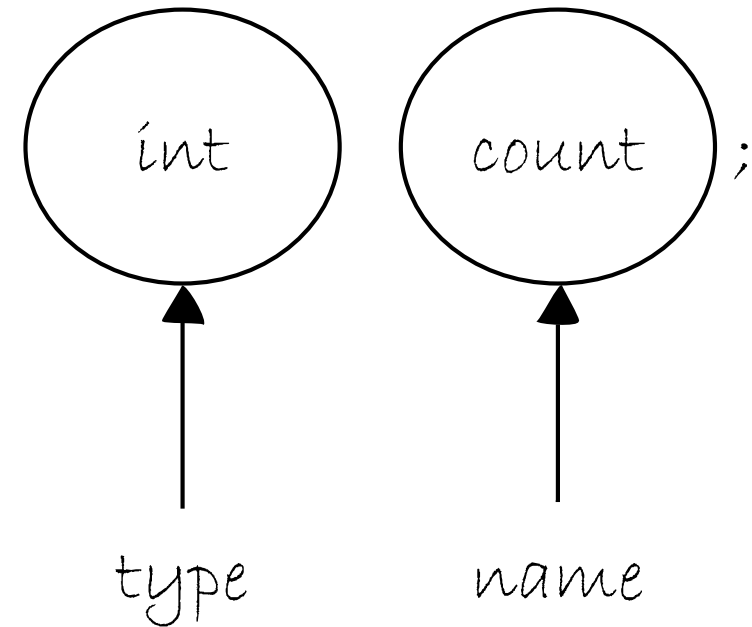


fig. 4.4

TIPI DI VARIABILE

- boolean: true or false
- char: un carattere , 'a', 'b', 'c', etc.
- String: una sequenza di caratteri
- byte: un numero piccolo, -128 to 127
- short: un numero un poco più grande, -32768 to 32767
- int: un numero intero, -2147483648 to 2147483647
- long: un numero estremamente grande
- float: numero con cifre decimali, such as 3.14159
- double: numero con tante cifre decimali



ASSEGNAZIONE DI UN VALORE

- La dichiarazione delle variabili avviene in maniera analoga a Javascript, ma invece della parola chiave var si usa il tipo adatto

```
int count;  
count = 50;
```

- Oppure
String nome = "Francesca";



ESEMPIO

Example 4-1: Variable declaration and initialization examples

```
int count = 0;           // Declare an int named count, assigned the value 0
char letter = 'a';       // Declare a char named letter, assigned the value 'a'
double d = 132.32;       // Declare a double named d, assigned the value 132.32
boolean happy = false;   // Declare a boolean named happy, assigned the value false
float x = 4.0;            // Declare a float named x, assigned the value 4.0
float y;                 // Declare a float named y (no assignment)
y = x + 5.2;             // Assign the value of x plus 5.2 to the previously declared y
float z = x*y + 15.0;     // Declare a variable named z, assign it the value which
                        // is x times y plus 15.0.
```


USO DI VARIABILI

// tra poco aggiungeremo le
nostre variabili

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

```
void draw(){  
    background(255);  
    stroke(0);  
    fill(175);  
    ellipse(100,100,50,50);  
}
```



USO DI VARIABILI (2)

// tra poco aggiungeremo le nostre
variabili

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

```
void draw(){  
    background(255);  
    stroke(0);  
    fill(175);  
    ellipse(mouseX,mouseY,50,50);  
}
```



USO DI VARIABILI (3)

```
// dichiariamo le nostre variabili
int circleX = 100;
int circleY = 100;

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

void draw(){
    background(255);
    stroke(0);
    fill(175);
    ellipse(circleX,circleY,50,50);
}
```



ASSEGNAZIONE DI UN NUOVO VALORE

// è possibile cambiare il valore alle variabili

```
int circleX = 0;  
int circleY = 100;
```

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

```
void draw(){  
  background(255);  
  stroke(0);  
  fill(175);  
  ellipse(circleX,circleY,50,50);  
  
  circleX = circleX + 1;  
}
```

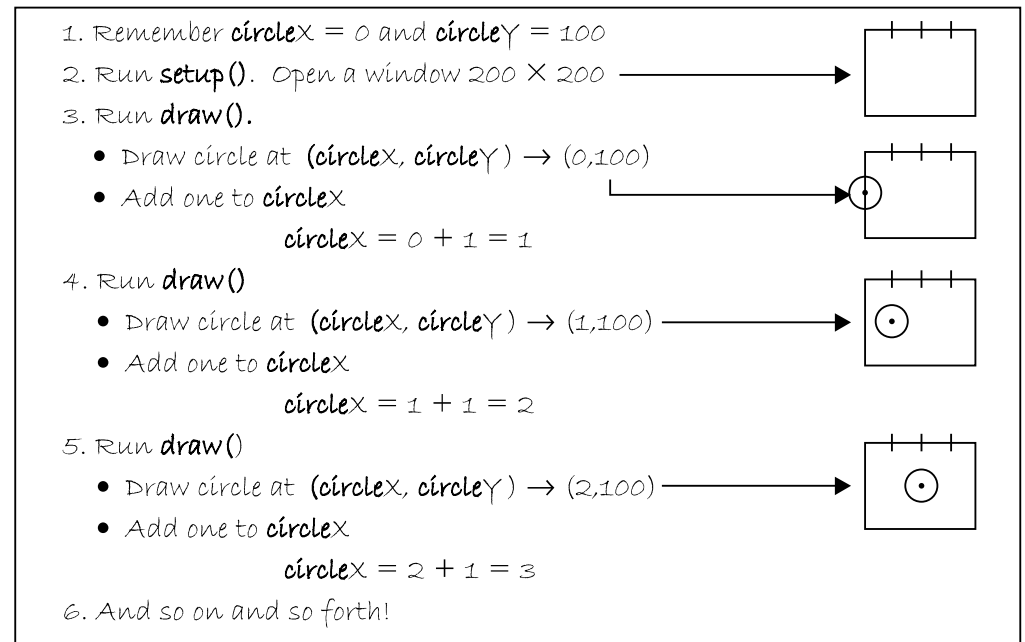


fig. 4.5

ESERCIZIO

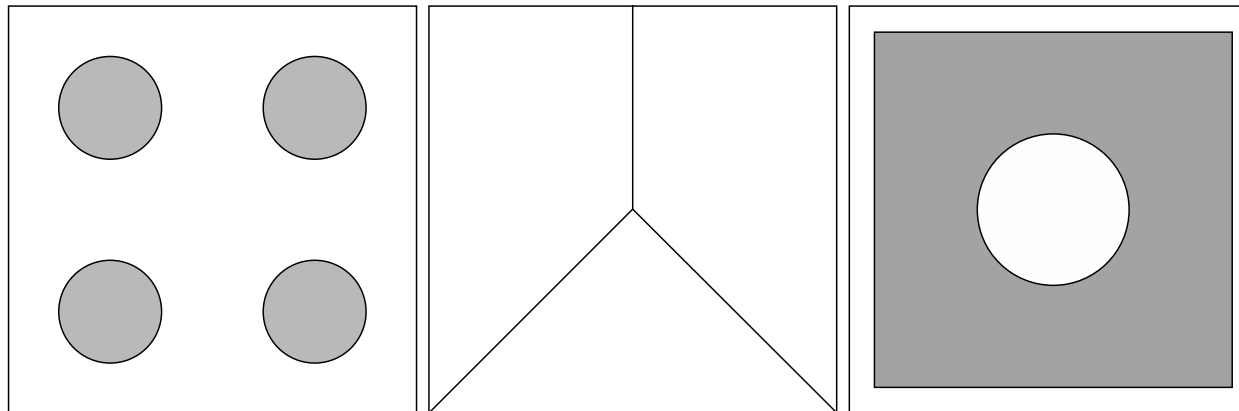
- Modifica l'esempio precedente per fare in modo che il cerchio cresca di dimensione



ESERCIZIO

Exercise 4-4

- Step 1:** Write code that draws the following screenshots with hard-coded values. (Feel free to use colors instead of grayscale.)
- Step 2:** Replace all of the hard-coded numbers with variables.
- Step 3:** Write assignment operations in **draw()** that change the value of the variables. For example, “`variable1 = variable1 + 2;`”. Try different expressions and see what happens!



VARIABILI DI SISTEMA

- Processing crea delle variabili prima dell'inizio della esecuzione
- Abbiamo già visto un esempio con mouseX e mouseY
- Ecco un elenco di variabili utili:
 - width: la larghezza della finestra
 - height: la altezza della finestra
 - frameCount: il numero di refresh della finestra dall'inizio dell'esecuzione
 - frameRate: numero di frame disegnati al secondo
 - screen.width: larghezza dello schermo
 - screen.height: altezza dello schermo
 - key: codice dell'ultimo tasto premuto
 - keyPressed: valore boolean, true se viene premuto un tasto



ESEMPIO

Example 4-5: Using system variables

```
void setup() {  
  size(200,200);  
  frameRate(30);  
}  
  
void draw() {  
  background(100);  
  stroke(255);  
  fill(frameCount/2);  
  rectMode(CENTER);  
  rect(width/2,height/2,mouseX+10,mouseY+10);  
}  
  
void keyPressed() {  
  println(key);  
}
```

frameCount is used to color a rectangle.

The rectangle will always be in the middle of the window if it is located at (width/2, height/2).

- Le variabili di sistema sono molto utili per:
 - ottenere valori che altrimenti non potreste avere (per esempio la posizione del mouse)
 - far funzionare il vostro codice in modo indipendente dai valori settati in precedenza (es: width in questo caso è 200, ma se cambiassi la dimensione della finestra automaticamente cambierebbe anche width)