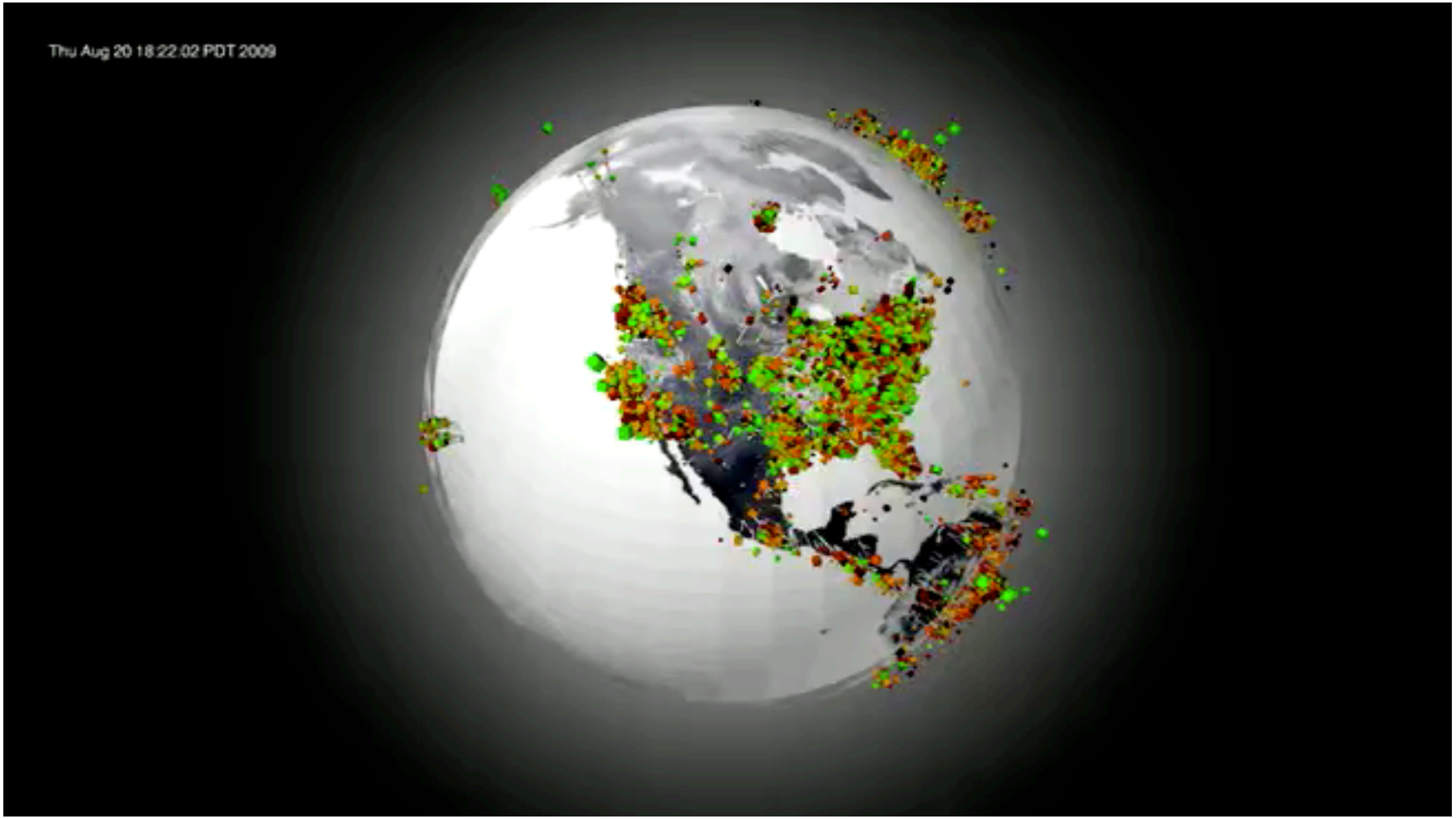
Introduzione a Processing

Roberto Ranon
www.dimi.uniud.it/ranon/processing.html

- Processing è, insieme, un ambiente e linguaggio di programmazione per creare applicazioni multimediali interattive, 2D e 3D
  - open source e gratuito*
  - basato su Java
  - per Linux, Mac OS X, e Windows
  - creato nel 2001 al MIT Media Lab, oggi utilizzato da migliaia di persone
  - il linguaggio per la programmazione di Arduino

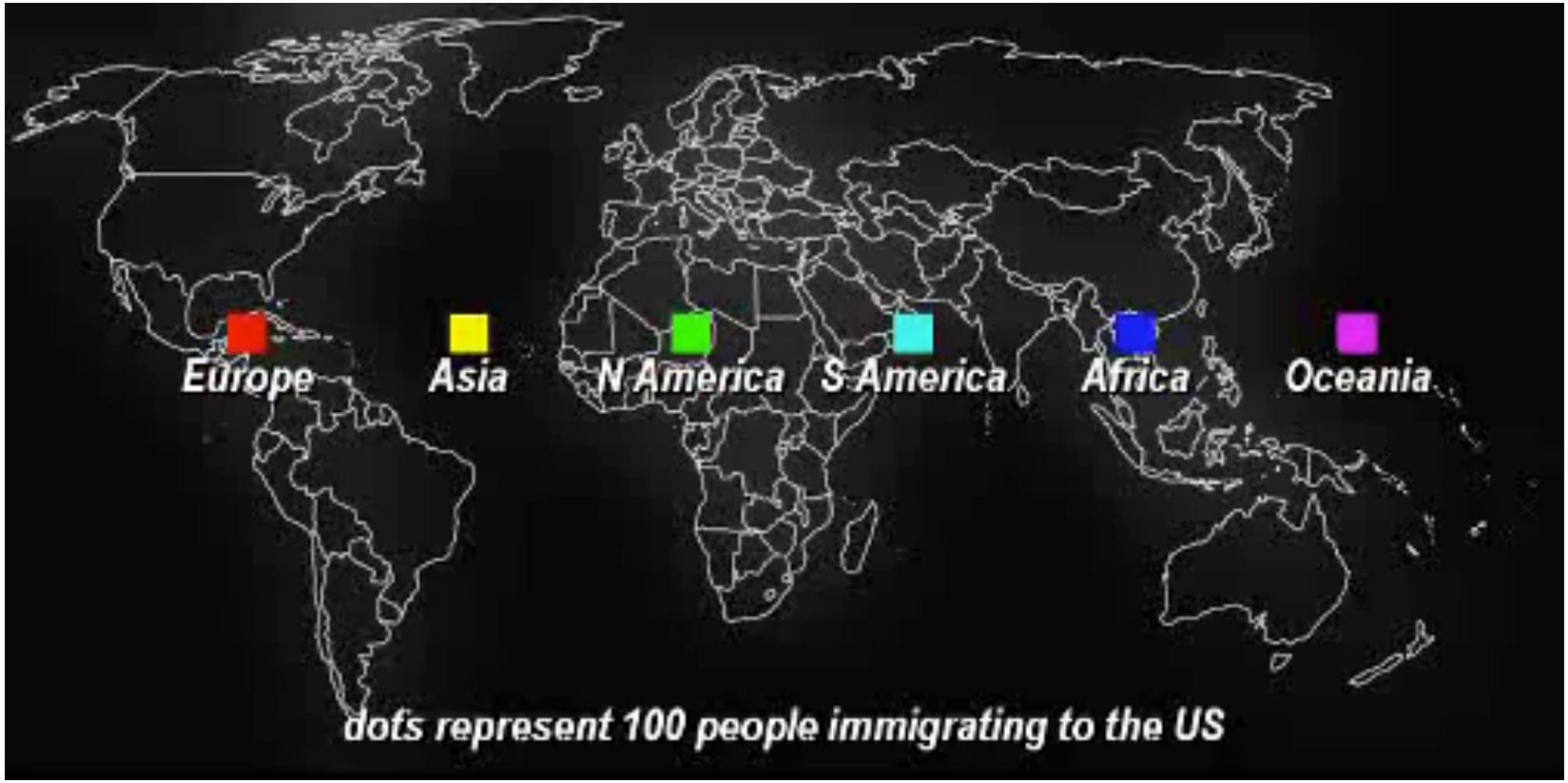
* processing.org/download/

Video Musicale per "**The Nest That Sailed The Sky**" di *Peter Gabriel*
realizzato con Processing da Glenn Marshall
https://vimeo.com/3245120



Good Morning! di *blprnt*

Una visualizzazione di 11,000 tweet raccolti nell'arco di 24 ore e contenenti le parole "good morning"

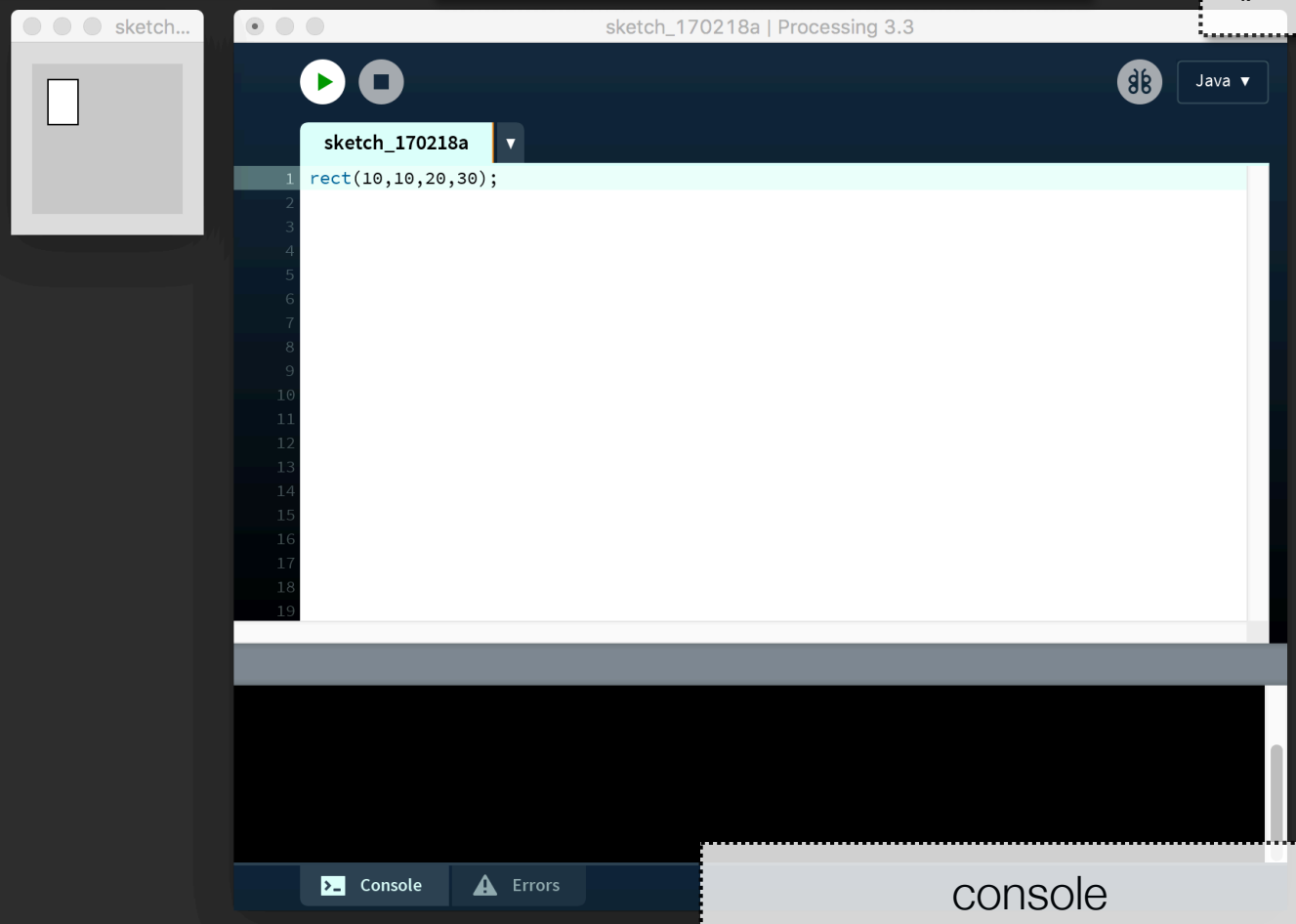


Immigrazione negli Stati Uniti 1820-2007

programma (sketch) in esecuzione

editor di codice processing

modalità di output (programma Java)



console

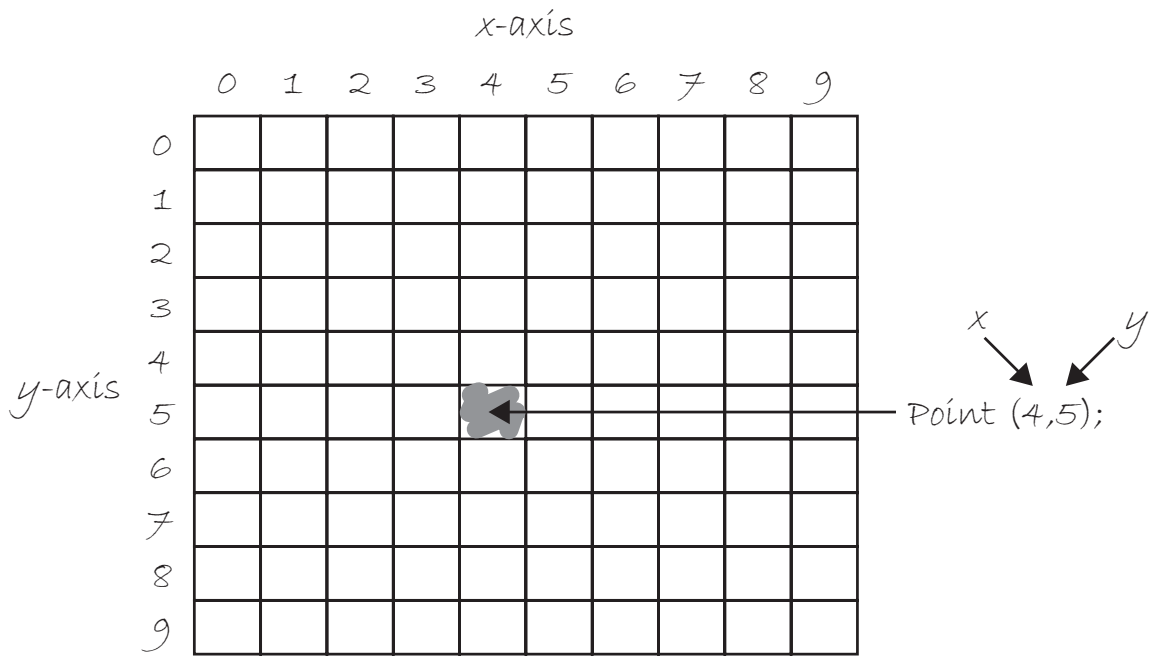
Processing Development Environment (PDE) versione 3.x

- i programmi Processing si chiamano *sketch,* e sono composti da:
  - una cartella su disco, col nome dello sketch, contenente …
  - … un file .pde, col nome dello sketch
  - eventualmente, altri file .pde e altre risorse all'interno della cartella
- i programmi possono essere esportati come applicazioni o pagine Web*

* con alcune limitazioni

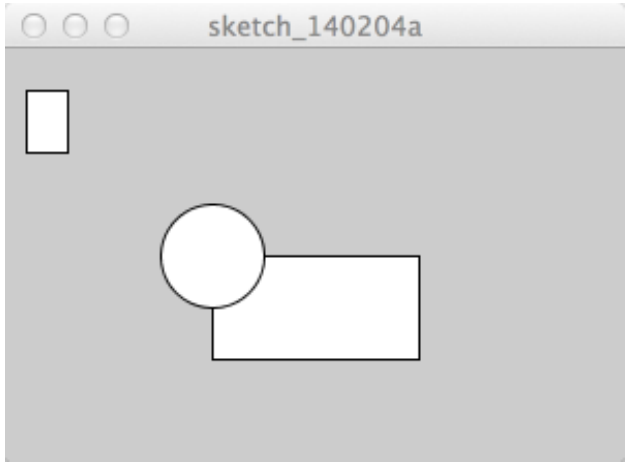
Forme e Colori

- *rect* è una delle istruzioni che *disegnano*
  - gli argomenti di questo tipo di istruzioni sono, di solito, in pixel
  - ogni pixel nella finestra di un'applicazione Processing è individuato dalle sue coordinate *x* e *y*:
    - *(0,0)* è il pixel in alto a sinistra
    - *(width-1, height-1)* è il pixel in basso a destra; *width* e *height* sono variabili di sistema

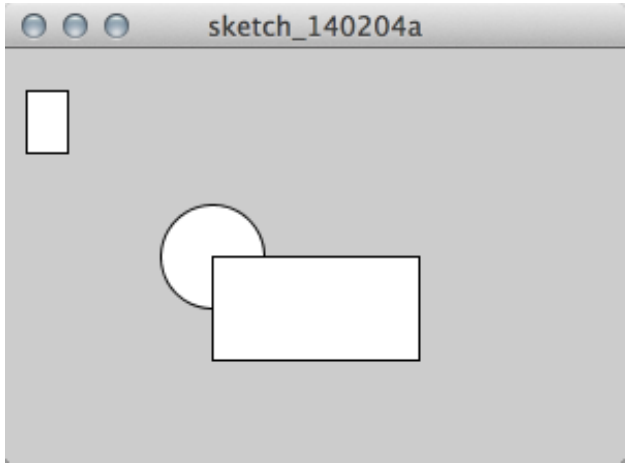


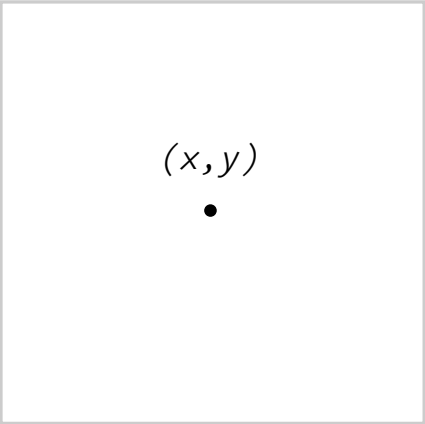
L’ordine delle istruzioni è, ovviamente, importante!

```
size(300,200);
rect(10,20,20,30);
rect(100,100,100,50);
ellipse(100,100,50,50);
```

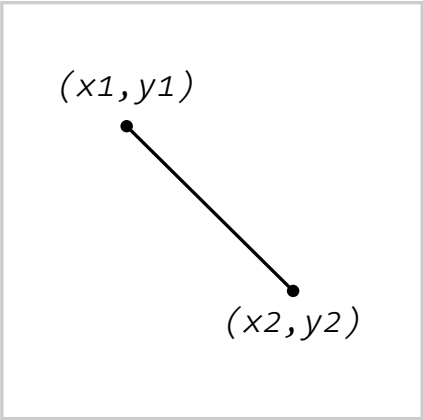



```
size(300,200);
rect(10,20,20,30);
ellipse(100,100,50,50);
rect(100,100,100,50);
```

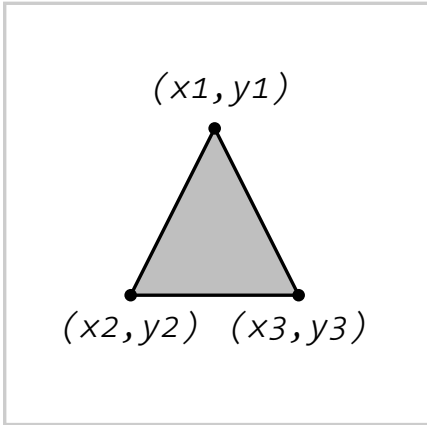




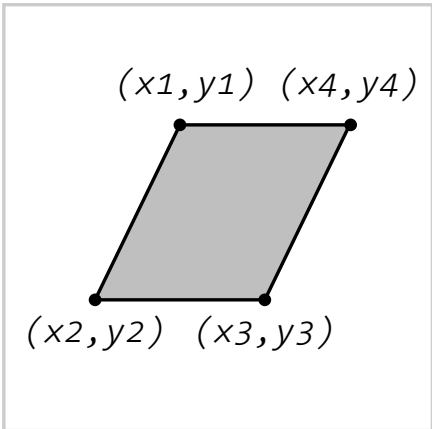
point(x,y)



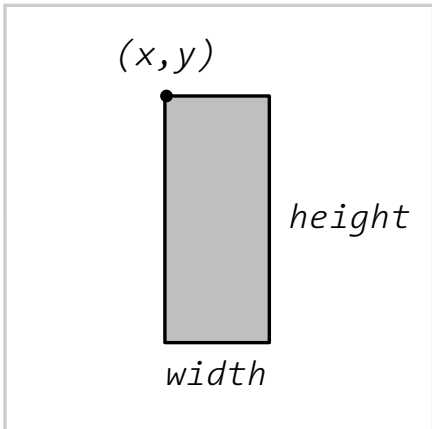
line(x1,y1,x2,y2)



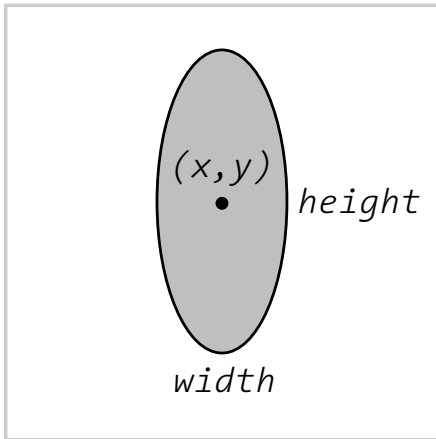
triangle(x1,y1,x2,y2,x3,y3)



quad(x1,y1,x2,y2,x3,y3,x4,y4)



rect(x,y,width,height)
rectMode(CORNER)



ellipse(x,y,width,height)
ellipseMode(CENTER)

- colori:
  - in scala di grigi: un valore in [0,255]
    - `background(255);  // Setting the background to white`
    - `stroke(0);  // Setting the outline (stroke) to black`
    - `fill(150);  // Setting the interior of a shape (fill) to grey`
    - `rect(50,50,75,100); // Drawing the rectangle`
  - RGB: three values in [0,255]
    - `fill(127,0,0);  // Dark red`
    - `ellipse(40,20,16,16);`
    - `fill(255,200,200);  // Pink (pale red)`
    - `ellipse(60,20,16,16);`
  - in scala di grigi, o RGB, più alfa
    - `fill(0,0,255);  // No fourth argument means 100% opacity`
    - `rect(0,0,100,200);`
    - `fill(255,0,0,255); // 255 means 100% opacity`
    - `rect(0,0,200,40);`
    - `fill(255,0,0,191); // 75% opacity`
    - `rect(0,50,200,40);`

- `stroke(color)`, `strokeWeight(width)`, `nostroke()`
  - controllano il colore con cui viene disegnato il contorno delle figure, lo spessore (in pixel), e la presenza del contorno
- `fill(color)`, `nofill()`
  - controllano il colore con cui viene disegnato l'interno delle figure, o la presenza dell'interno
- **importante**: queste istruzioni impostano lo stato che viene utilizzato per disegnare, che resta attivo finché altre istruzioni lo modificheranno

Animazioni e Interattività

```
void setup()
{
 size(500,300);
}
```

eseguito all'avvio del programma

```
void draw ()
{
  // Displays the frame count to the Console
  println("I'm drawing");
  println(frameCount);
}
```

eseguito dopo setup(). Il codice all'interno di draw() viene eseguito ciclicamente fino alla chiusura del programma

```
int diam = 20;

void setup()
{
 size(500,300);
 background(180);
 stroke(0);
 strokeWeight(3);
 fill(255,50);
}

void draw ()
{
  ellipse( width/2, height/2, diam, diam);
}
```

alternativa 1

```
void draw ()
{
  if (diam < 500) diam++;
  ellipse( width/2, height/2, diam, diam);
}
```

alternativa 2

```
void draw ()
{
  background(180);
  if (diam < 500) diam++;
  ellipse( width/2, height/2, diam, diam);
}
```

alternativa 3

```
void draw ()
{
  ellipse( mouseX, mouseY, diam, diam);
}
```

- un programma può essere una lista di istruzioni, o di funzioni;
  - nel primo caso, le istruzioni vengono eseguite, e il programma si ferma;
  - nel secondo case, possiamo scrivere un programma interattivo tramite le due funzioni setup() e draw()
- la sintassi, i tipi di dato di base, gli operatori, … funzionano esattamente come in Java

```
void setup()
{
  size(500,300);
  background(180);
  stroke(0);
  strokeWeight(3);
}

void draw()
{
   fill(random(0,255));
   ellipse ( random( width), random( height ), random( 50 ), random( 50 ));
}
```

random(n) genera un numero casuale tra 0 e n (non incluso)
random(low, high) genera un numero casuale tra low e high (non incluso)