

# Corso di **Sistemi Interattivi**

## **Lezione 4. Processing e la Grafica**

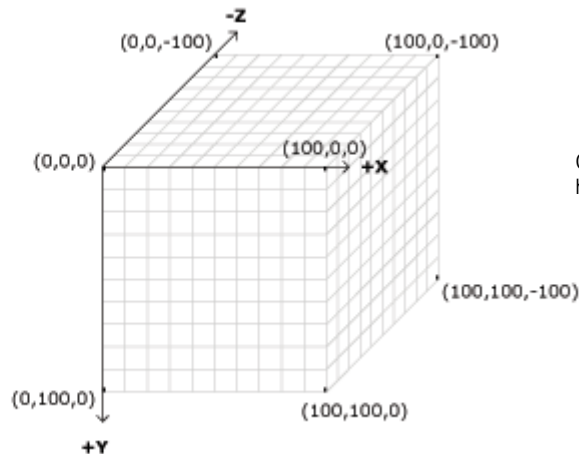
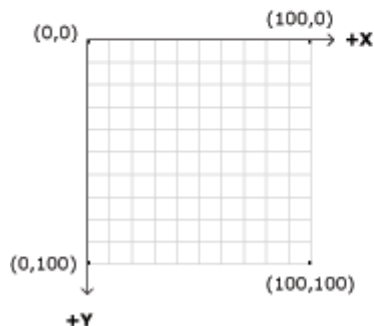
Prof. Rudy Melli ([rudymelli@ababrera.it](mailto:rudymelli@ababrera.it))

[www.vision-e.it/si](http://www.vision-e.it/si)

**ACCADEMIA DI BELLE ARTI DI BRERA**  
**Anno accademico 2019/2020**

# Il modo grafico di Processing

- Di default è una finestra 2D ma può essere settato per lavorare in 3D
- In 2D la finestra grafica è come un sistema cartesiano dove l'origine è il primo punto in alto a sinistra, la x cresce verso destra e la y cresce verso il basso:



Credits:  
<https://processing.org/reference/environment/>

- L'unità di misura è il pixel

# Finestra di render

## *dentro funzione setup() o settings()*

- ▣ Dimensione finestra → *size(width, height)* *size(width, height, renderMode)*
  - ☞ *renderMode* definisce il 2D o il 3D e può essere P2D o P3D
  - ☞ NB: in Processing 3.0 è cambiato il comportamento di *size*, se deve essere definito con dimensioni derivanti da variabili, è necessario spostarlo in una nuova funzione di sistema (tipo *setup()*) chiamata ***settings()*** che viene chiamata prima di fare qualsiasi cosa, anche prima di *setup()*
- ▣ Impostare il framerate di rendering → *frameRate(fps)*
- ▣ Fullscreen → *fullScreen(display)* *fullScreen(renderMode, display)*
  - ☞ Display è il numero di schermo (1, 2, 3, ...) su cui far partire in fullscreen la patch oppure inserendo SPAN va in multischermo
  - ☞ *renderMode* vedi sopra
  - ☞ *fullScreen* è da usare come alternativa a *size()*
- ▣ Nascondere il puntatore del mouse → *noCursor()*
  
- ▣ NB: usare il Reference (<https://processing.org/reference/>) per gli approfondimenti

# Variabili di sistema

Sono variabili settate dal sistema con le proprie informazioni che possono essere utilizzate nel programma (vengono colorate in fucsia nel codice)

- ▣ Dimensione finestra → *width, height*
- ▣ Framerate → *frameRate*
- ▣ Numero di frame di Run → *frameCount*
  - ☛ Numero che parte da 0 ed incrementa di 1 ogni volta che viene eseguito *draw()*. E' Utile per sapere a che punto si è durante l'esecuzione e, ad esempio, cambiare qualcosa ad ogni frame
- ▣ La finestra é in primo piano, quindi sta rilevando mouse e tastiera? → *focused*
- ▣ Coordinate del mouse → *mouseX, mouseY*
- ▣ E' stato premuto un pulsante del mouse? → *mousePressed*
- ▣ Quale pulsante del mouse premuto → *mouseButton*
- ▣ E' stato premuto un tasto della tastiera? → *keyPressed*
- ▣ Quale tasto della tastiera è stato premuto → *key, keyCode*
  - ☛ *keyCode* identifica un tasto speciale (Su, Giu, Destra, Sinistra, Ctrl...), per capire se è stato premuto un tasto speciale usare *if(key == CODED)*

# Geometrie di base

- ❏ Punto → *point(x,y)*
- ❏ Linea → *line(x1, y1, x2, y2)*
- ❏ Triangolo → *triangle(x1, y1, x2, y2, x3, y3)*
- ❏ Cerchio/Ellisse → *ellipse(x, y, r1, r2)*
- ❏ Rettangolo → *rect(x1, y1, x2, y2)*
  
- ❏ Colore di sfondo → *background(grigio) background(r, g, b) background(r, g, b, alpha)*
- ❏ Colore di riempimento → *fill(grigio) fill(r, g, b) fill (r, g, b, alpha)*
- ❏ Colore di bordo → *stroke(grigio) stroke(r, g, b) stroke(r, g, b, alpha)*
  
- ❏ Rendering e geometrie 3D → <https://processing.org/tutorials/rendering/>

☞ NB: in caso di problemi di avvio in modalità PD\_3D, aggiornare i driver della scheda video

# Immagini

- Un'immagine è vista dal computer come una tabella o un diagramma cartesiano dove l'origine (0, 0) è nel primo punto in alto a sinistra, la x cresce verso destra e la y verso il basso. Ogni valore della tabella rappresenta il colore (se RGB) o la luminosità (se in scala di grigi) del pixel:

		x →				
		0	1	2	3	4
y	0	0	1	2	3	4
	1	5	6	7	8	9
	2	10	11	12	13	14
	3	15	16	17	18	19
	4	20	21	22	23	24
		← width = 5 →				

Pixel 13 has an x value of 3 and y value of 2.


$$\begin{aligned}
 &x + (y * \text{width}) \\
 &= 3 + (2 * 5) \\
 &= 3 + 10 \\
 &= 13
 \end{aligned}$$

- <https://processing.org/tutorials/pixels/>




# Immagini

- In Processing il tipo dati immagine è `PImage`
- Per caricare un'immagine si usa la funzione *`loadImage(nome file)`*
  - ☞ Es: *`PImage img = loadImage("moonwalk.jpg");`*
- Per visualizzare un'immagine nella finestra di rendering bisogna inserire la funzione
  - ☞ Visualizzare l'immagine alle coordinate x,y: *`image(immagine, x, y)`*
  - ☞ Visualizzare l'immagine alle coordinate x,y e con dimensioni width x height (in pixel): *`image(immagine, x, y, width, height)`*
  - ☞ E' bene posizionarla dentro la funzione `draw()` per renderizzare l'immagine costantemente
  - ☞ Per conoscere le dimensioni di una variabile immagine si utilizzano due variabili/proprietà che si raggiungono aggiungendo il punto al nome dell'immagine:
    - `img.width`
    - `img.height`
  - ☞ Esempio: `Basic`→`Image`→`LoadDisplayImage`
- Per modificare il canale alpha (trasparenza) → *`tint(255, alpha)`*
  - ☞ Esempio `VideoMixer`











# Reference PImage

 PImage non è un tipo dati semplice, ma è una classe. Aggiungendo il punto alla fine del nome è possibile utilizzare proprietà e funzioni relative a quella specifica immagine

## Fields:

-  pixels[] Array containing the color of every pixel in the image
-  width Image width
-  height Image height

## Methods:

-  loadPixels() Loads the pixel data for the image into its pixels[] array
-  updatePixels() Updates the image with the data in its pixels[] array
-  resize() Changes the size of an image to a new width and height
-  get() Reads the color of any pixel or grabs a rectangle of pixels
-  set() writes a color to any pixel or writes an image into another
-  mask() Masks part of an image with another image as an alpha channel
-  filter() Converts the image to grayscale or black and white
-  copy() Copies the entire image
-  blend() Copies a pixel or rectangle of pixels using different blending modes
-  save() Saves the image to a TIFF, TARGA, PNG, or JPEG file



# Libreria processing.video

- Si installa dall'interfaccia di Processing
  - ☛ [http://interattivi.altervista.org/si\\_wordpress/processing-librerie-aggiungere-installarle/](http://interattivi.altervista.org/si_wordpress/processing-librerie-aggiungere-installarle/)
- Permette di caricare filmati e di usare webcam e dispositivi video di input
- Una volta installato trovi gli esempi in
  - ☛ Libraries → Video
- Per utilizzarla dentro uno sketch è necessario inserire all'inizio la seguente istruzione:
  - ☛ `import processing.video.*;`

# Filmati

- ❏ Esempio Libraries→Video→Movie→Loop
- ❏ Bisogna sempre caricare la libreria all'inizio dello sketch con l'istruzione:
  - ☞ `import processing.video.*;`
- ❏ Il nome del filmato si specifica nella creazione della variabile di tipo `Movie`:
  - ☞ `movie = new Movie(this, filmato);`
- ❏ La funzione `movieEvent(Movie m)` viene eseguita dal sistema con una cadenza pari al framerate del filmato, la variabile `Movie m` è il filmato
- ❏ `m.read()` legge il frame corrente del filmato
- ❏ `if(movie.available() == true){movie.read();}` è alternativo all'utilizzo di `movieEvent`
- ❏ Il frame viene visualizzato sempre con la funzione `image` utilizzata per le immagini a cui viene passata la variabile `Movie` invece che la singola immagine
  - ☞ Visualizzerà il frame caricato da `.read()`

```

8 import processing.video.*;
9
10 Movie movie;
11
12 void setup() {
13   size(640, 360);
14   background(0);
15   // Load and play the video in a loop
16   movie = new Movie(this, "transit.mov");
17   movie.loop();
18 }
19
20 void movieEvent(Movie m) {
21   m.read();
22 }
23
24 void draw() {
25   //if (movie.available() == true) {
26   //  movie.read();
27   //}
28   image(movie, 0, 0, width, height);
29 }
30

```

# WebCam

## ❏ Esempio SimpleCapture

## ❏ Bisogna sempre caricare la libreria all'inizio dello sketch con l'istruzione:

☞ `import processing.video.*;`

## ❏ Se non viene specificato il nome della webcam viene utilizzata la prima disponibile

☞ `cam = new Capture(this, 640, 480);`

Altrimenti

☞ `Cam = new Capture(this, width, height, nome, framerate);`

## ❏ La webcam deve essere fatta partire con `cam.start();`

## ❏ `if(cam.available() == true)` notifica un nuovo frame pronto dalla webcam e lo carica con `{cam.read();}`

## ❏ Il frame viene visualizzato sempre con la funzione `image` utilizzata per le immagini a cui viene passata la variabile `Capture` invece che la singola immagine

☞ Visualizzerà il frame caricato da `.read()`

```

1 import processing.video.*;
2
3 Capture cam;
4
5 void setup() {
6   size(640, 480);
7   printArray(Capture.list());
8   cam = new Capture(this, 640, 480);
9   // Or, the settings can be defined based on the text
10  //cam = new Capture(this, 640, 480, "Built-in iSight");
11  // Start capturing the images from the camera
12  cam.start();
13 }
14
15 void draw() {
16   if (cam.available() == true) {
17     cam.read();
18   }
19   image(cam, 0, 0, width, height);
20   // The following does the same as the above image() but
21   // is faster when just drawing the image without any
22   // resizing, transformations, or tint.
23   //set(0, 0, cam);
24 }
```

# WebCam

🖥️ Esempio Libraries→Video→Capture→GettingStartedCapture

🖥️ Qui viene stampata nella finestra dei messaggi la lista delle webcam connesse (cameras) ed aperta la prima della lista:

- ☞ `String[] cameras = Capture.list();`
- ☞ `cam= new Capture(this, cameras[0]);`

```

7 import processing.video.*;
8
9 Capture cam;
10
11 void setup() {
12     size(640, 480);
13     String[] cameras = Capture.list();
14     if (cameras == null) {
15         println("Failed to retrieve the list of available cameras");
16         cam = new Capture(this, 640, 480);
17     } if (cameras.length == 0) {
18         println("There are no cameras available for capture");
19         exit();
20     } else {
21         println("Available cameras:");
22         printArray(cameras);
23         // The camera can be initialized directly using the constructor
24         // from the array returned by list():
25         cam = new Capture(this, cameras[0]);
26         // Or, the settings can be defined based on the camera's name:
27         // cam = new Capture(this, 640, 480, "Built-in Webcam");
28         // Start capturing the images from the camera
29         cam.start();
30     }
31 }
32
33 void draw() {
34     if (cam.available() == true) {
35         cam.read();
36     }
37     image(cam, 0, 0, width, height);
38     // The following does the same as the above but
39     // is faster when just drawing the image without
40     // resizing, transformations, or tint.
41     //set(0, 0, cam);
42 }

```

# Esercizi

- Aprire le 3 immagini presenti nella cartella condivisa aa1920\_si/media/seq3/ e visualizzarle in una sequenza a loop, ad ogni frame visualizzare un immagine diversa. Impostare il framerate a 3fps
- Aprire un video (con audio) e visualizzarlo in fullscreen a velocità doppia (vedi: <https://processing.org/reference/libraries/video/index.html>)
- Modificare la velocità aumentando/diminuendo del 20% con i tasti “+” e “-” della tastiera
- Visualizzare la webcam nella finestra di rendering in alto a destra in una porzione che occupa un quarto delle dimensioni della finestra
- Cambiare il framerate della webcam riducendolo a 1fps
- Alternare all’immagine della webcam o di un filmato, un frame completamente nero
- Testare gli esempi:
  - ☞ *Topics/Image Processing*
  - ☞ *Libraries/Video*