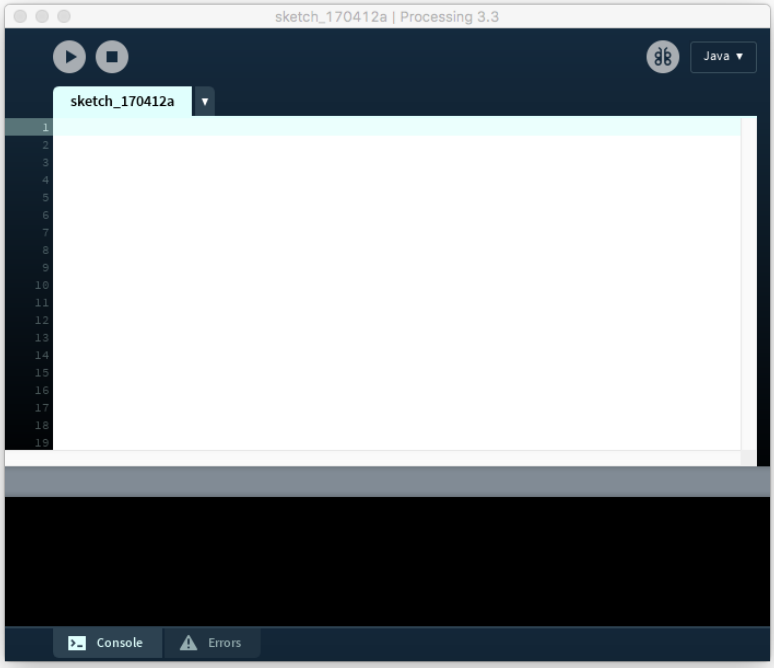
Programmare in Processing - Parte 1

Perché processing?

- Processing è progettato per rendere facile creare programmi visuali e interattivi senza richiedere molte linee di codice. Questo rende l'uso di Processing facile e divertente.
- Processing nasce di fatto come una variante di Java ed è facile portare codice Processing verso Javascript tramite l'uso di librerie dedicate (Processing.js or p5.js). Per questo può essere utilizzato come ponte verso questi linguaggi più diffusi: l'idea è imparare ad utilizzare Processing e poi passare a questi altri linguaggi.

Come installarlo

Per installare Processing scaricare il pacchetto adatto per il proprio sistema operativo dal sito del progetto alla pagina: https://processing.org/download/



Una volta lanciato il programma apparirà la finestra dell'editor di codice, che è dove scriveremo il nostro codice.

Proviamo a scrivere il nostro primo programma. Disegniamo un cerchio, scrivendo all'interno della finestra di Processing la seguente linea di codice:

```
ellipse (50, 50, 75, 75);
```

Clicchiamo sul pulsante con il triangolo in alto a sinistra e otterremo una finestra con un cerchio disegnato.

Funzioni

In processing, come in molti altri linguaggi di programmazione, uno dei principali metodi utilizzati per ottenere risultati è **chiamare funzioni**.

- *Funzioni = Procedure = "crea un blocco" in Scratch*

In generale un programma è un insieme di istruzioni che dicono al computer di fare qualcosa seguendo una sequenza di passi.
Una funzione è **uno di questi passi**. Chiamare una funzione corrisponde a dare al computer una **singola istruzione** per fare una **sola cosa**.

Come si chiama una **funzione** in processing?

1. Scrivere il nome della funzione
2. Aggiungere una coppia di parentesi tonde **( )** dopo il nome della funzione
3. Inserire all'interno delle parentesi i **parametri** richiesti dalla funzione. I parametri devono essere separati da virgole
4. Terminare la linea con un punto e virgola ---> **;**

Funzioni

I computer sono "stupidi". Se gli vogliamo dire di fare un salto, dobbiamo anche dirgli esattamente quanto lontano saltare. Queste informazioni vengono fornite tramite i **parametri**.
Per esempio, supponiamo di voler fornire delle istruzioni per guidare. Col codice potremmo farlo con questa sequenza di istruzioni:

```
drive(5);
turnLeft();
drive(3);
```

(notare che turnLeft() non necessita di parametri per cui non ne vengono forniti)

Torniamo adesso all'istruzione che abbiamo visto in precedenza:

```
ellipse (50, 50, 75, 75);
```

ellipse() è una funzione fornita da Processing per disegnare cerchi ed ellissi. Utilizza quattro parametri: La posizione del centro con una coppia di coordinate **x** e **y**, una **larghezza** e una **altezza**.

Proviamo a sperimentare con il codice:

- Che valore dare ai parametri se volessimo disegnare un cerchio di diametro 20 e con centro alle coordinate x = 50 e y = 75?
- Come disegnare un'ellisse centrata nell'angolo in alto a sinistra o in basso a destra?
- Come disegnare un cerchio che riempie completamente la finestra?
- Come disegnare ellisse molto schiacciate orizzontalmente o verticalmente?

Processing mette a disposizione molte altre funzioni. Per avere informazioni su come usarle si possono utilizzare le pagine con il manuale di riferimento al seguente URL:

https://processing.org/reference/

Questa pagina contiene informazioni su tutte le funzioni fornite da Processing.
Cliccando su una particolare funzione si ottengono i dettagli su quello che fa e su come utilizzarla (per esempio di che parametri ha bisogno).

Per esempio, troviamo una funzione per disegnare rettangoli.
Questa è la funzione **rect()** e ha lo stesso tipo di parametri della funzione **ellipse()**: le coordinate x,y questa volta dell'angolo in alto a sinistra, la larghezza e l'altezza.
Possiamo quindi modificare il programma per disegnare un rettangolo invece di un cerchio. Per esempio:

```
rect (10, 20, 80, 70);
```

Siamo già programmatori ;)

Abbiamo capito che per risolvere un problema con il computer dobbiamo "semplicemente" consultare la guida, trovare le istruzioni che fanno quello che ci servono, e scrivere il codice per testare che il tutto funzioni. Questo è quanto fanno i programmatori per il 95% del tempo.
Per il resto si tratta solamente di trovare problemi da risolvere e scrivere il codice che serve per risolverli.

Per esempio, come possiamo aumentare la dimensione della finestra dove viene eseguito il nostro programma?
Cercando nella guida troveremo la funzione **size()** che ha due parametri: larghezza e altezza. Ecco quindi come fare per visualizzare una ellisse più grossa:

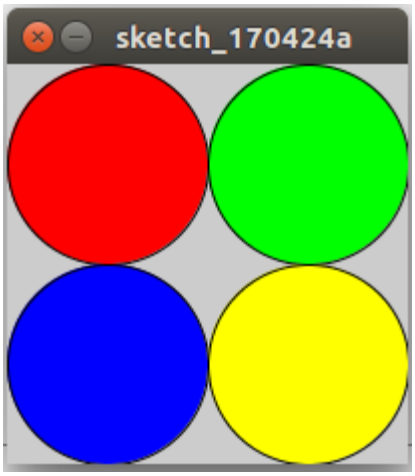
```
size(500, 300);
ellipse(250, 150, 300, 100);
```

Se vogliamo cambiare il colore del nostro cerchio possiamo utilizzare la funzione **fill()** che prende tre parametri: il livello di rosso, quello di verde e quello di blu (i livelli vanno da un minimo di 0 a un massimo di 255). Per disegnare una ellisse rossa basta aggiungere la seguente 'istruzione prima di chiamare **ellipse()**:

```
fill(255, 0, 0);
```

Ecco un programma che disegna 4 cerchi di differenti colori:

```
size(200, 200);
fill(255, 0, 0);
ellipse(50, 50, 100, 100);
fill(0, 255, 0);
ellipse(150, 50, 100, 100);
fill(0, 0, 255);
ellipse(50, 150, 100, 100);
fill(255, 255, 0);
ellipse(150, 150, 100, 100);
```



ESERCIZI

Il miglior modo di imparare a programmare consiste nello scrivere codice. Diamoci dei compiti e cerchiamo di risolverli consultando la guida di processing alla ricerca di funzioni che possono aiutarci. Ecco alcuni esempi:

1. Disegnare una faccia sorridente
2. Disegnare un fiore o un giardino
3. Disegnare un arcobaleno
4. Disegnare un cane o un gatto
5. Disegnare una casa

Soluzioni

Disegnare una faccia sorridente:

```
ellipse(50, 50, 75, 75);
fill(0, 0, 0);
ellipse(50-15, 50-15, 10, 10);
ellipse(50+15, 50-15, 10, 10);
noFill();
arc(50, 60, 40, 30, 0, PI);
```

Disegnare un fiore

Tbc

Disegnare un arcobaleno

Tbc

Disegnare un cane o un gatto

Tbc

Disegnare una casa

Tbc