**PROCESSING**

**Lezione 1**

//size(500, 500); //dimenzione tavola

//Figure disponibili:

//circle(250, 150, 250); // (xc, yc, diametro)

//rect(0, height/2-h/2, width, h); //xr, yr, lunghezza e altezza

//triangle(width/2, 0, width, height, 0, height); //cordinate dei tre punti

//ellipse(250, 150, 300, 500); // (xc, yc, diametro orizontale, diametro verticale)

//point(255, 255); // cordinate del punto

//line(0, 0, width, height); //cordinate del primo punto e del secondo

/\*size(500, 500);

ellipseMode(CORNERS);

ellipse(100, 100, 200, 200);

\*/

/\*Constanti con ellissi:

center=terzo e quarto paremtro sono il diametro(default)

radius =terzo e quarto parametro sono raggio orizontale e verticale

corner =interpreta i primi due parametri di ellipse() come l'angolo superiore

sinistro della forma, mentre il terzo e il quarto parametro sono la larghezza

e l'altezza.

corners = interpreta i primi due parametri di ellipse() come la posizione

di un angolo del rettangolo di selezione dell'ellisse e il terzo e il

quarto parametro come la posizione dell'angolo opposto.

//ellipseMode(CENTER);

//ellipseMode(RADIUS);

//ellipseMode(CORNER);

//ellipseMode(CORNERS);

Con width e height posso riempire tutta la finsetra

\*/

//Possiamo modificare il colore cosi:

//background(0, 0, 0); // sfondo, i parametri per i colori

//colorMode(RGB, 1.0); // ridefinire la scala

//stroke(0, 0, 0); // colore controrno

// (10); // spessore contorno

//fill(255, 255, 0); // colore figura interna

//fill(#0FA008); // con codice colore

//noStroke(); // toglie i bordi

// inoltre possiamo usare random per selezionale colori randomici

/\* possiamo usare cicli for e variabili e anche gli operatori logici\*/

println("valore di i: "); // per stampare

/\*

la funzione setup(){} viene eseguite solo una volta.

la funzione draw(){} è un loop infinito (for(){} è finito) e viene richiamato

60 volte al secondo (60fps).

quando si definiscono queste funzioni, si devono creare promettendo il tipo di dato

che restituiscono (ad esempio random() restituisce un float, color() restituisce un colore),

nel caso di queste due funzioni, non restituiendo nessun dato, saranno accompagnate da "void"

\*/

void setup(){

println("sono nel setup");

frameRate(2); //la funzione frameRate() definisce gli fps. di defaulf è a 60.

}

void draw(){

println("mi trovo nel draw al frame: "+frameCount); //frameCount è una variabile che conta i frame in cui ci si trova per ogni ciclo

}

**Lezione 2**

size(480,120);

//cambio il modo di interpretare la funzione ellipse

//figura1

rect(50, 60, 80, 80);

ellipse(50, 60, 80, 80);

//figura2

ellipseMode(CORNER); //le coordinate non sono più del centro ma il vertice in alto

//a sx del quadrato/rettangolo in cui la circonferenza è inscritta

rect(150, 20, 80, 80);

ellipse(150, 20, 80, 80);

//figura3

ellipseMode(CENTER); //ritorno all'interpretazione di default, con coordinate che indicano il centro dell'ellisse/cerchio, uguale alla figura 1

rect(290, 20, 80, 80);

ellipse(290, 20, 80, 80);

/\*

con la funzione "ellipseMode(RADIUS)" le coordinate sono del centro,

ma cambiano il terzo e quarto argomento che non indicano più la lunghezza del diametro/corda del cerchio/ellisse ma la lunghezza del suo raggio

\*/

//figura4 - differenza tra RADIUS (in nero) e CENTER (in bianco)

fill(0);

ellipseMode(RADIUS);

ellipse(430, 60, 40, 40);

fill(255);

ellipseMode(CENTER);

ellipse(430, 60, 40, 40);

/\*

con la funzione "ellipseMode(CORNERS)" viene definito il rettangolo che inscrive il cerchio.

le coordinate sono del vertice in alto a sx e del vertice in basso a dx.

processing non disegna il rettangolo ma solo il cerchio inscritto.\*/

/\*

le stesse funzioni CENTER, CORNER, CORNERS e RADIUS valgono per il rettangolo.

la differenza sta nella funzione di default, che per il cerchio è CENTER e per il rettangolo è CORNER

\*/

color c1= color(204,154,0); //Funzione per assegnare colore a c1

color c2= #FFCC00; //Colore esadecimale

noStroke();

fill(c1); //Riempimento

rect(0,0,25,100); //Rettangolo

fill(c2); //Riempimento

rect(25,0,25,100); //Rettangolo

color c3= get(10,50);

fill(c3);//Riempimento

rect(50,0,500,100);//Rettangolo

// Costanti rappresentati radianti (angoli da usare in arc

// PI= pi greco (180)

// HALF\_PI=Mezzo Pi greco (90)

// QUARTER\_PI=Un quarto di Pi greco (45)

// TWO\_PI=Due volte Pi greco (360)

//i primi 4 argomenti sono come in ellipse, gli ultimi due sono il punto

//di inizio e fine tra i quali viene tracciato l'arco (espressi in radianti)

size(480,120);

arc(90,60,80,80,0,HALF\_PI);

arc(190,60,80,80,0,PI+HALF\_PI);

arc(290,60,80,80,PI,TWO\_PI+HALF\_PI);

arc(390,60,80,80,QUARTER\_PI,PI+QUARTER\_PI);

size(480,120);

strokeWeight(24); //Spessore

line(60,25,130,95); // Linea

strokeCap(SQUARE); // Spessore quadrato

line(160,25,230,95); // Linea

strokeCap(PROJECT); // Spssore che continua

line(260,25,330,95);// Linea

strokeCap(ROUND); // Per ripristinare Default strokeCap

line(360,25,430,95);// Linea

// Cap = SQUARE;PROJECT;ROUND default ROUND

//strokejoin()

strokeJoin(ROUND);

rect(160, 25, 70, 70); //stroke arrotondato

strokeJoin(BEVEL);

rect(260, 25, 70, 70); //stroke "tagliato" alla punta

strokeJoin(MITER);

rect(360, 25, 70, 70); //ritorno allo stroke di default

beginShape();//unisce i punti forniti successivamente

fill(153,176,180); //colore di riempimento

vertex(180,82);

vertex(207,36);

vertex(214,63);

vertex(407,11);

vertex(412,30);

vertex(219,82);

vertex(226,109);

endShape();//la figura resta aperta

//endShape(CLOSE) chiude la figura (unisce l'ultimo vertice col primo)

Curva di beizer

bezier(185,50,10,50,15,290,190,300);

**Lezione 3**

void mousePressed() //tasto mouse

{

print("P");

}

void mouseReleased()

{

print("R");

}

void mouseClicked()

{

print("C");

}

void keyPressed() //tasto invio

{

print("Pk");

}

void keyReleased()

{

print("Rk");

}

void keyTyped() // tasto spaziatrice

{

print("Tk");

}

//Esiste la funzione mousepressed e relased

void mousePressed(){

noLoop();

}

void mouseReleased(){

loop();

}

//Possiamo eguagliare

if(mouseButton==LEFT){

loop();

}

if(mouseButton==RIGHT){

noLoop();

}

//Posso associace anche a tasti, inoltre posso usare anche valore boleani

if(keyPressed && (key==CODED)){//il codice associato a determinati tasti(non per le lettere)

if (keyCode==LEFT){

x--;

}

//è codificato non è in left ma in right se solo con else si attiverebbe con qualsiasi altro tasto

else if(keyCode==RIGHT){

x++;

}

//Lerp = Calcola un numero compreso tra due numeri con un incremento specifico

size(400, 400);

int x1 = 60;

int y1 = 40;

int x2 = 320;

int y2 = 360;

line(x1, y1, x2, y2);

for (int i = 0; i <= 40; i++) {

float x = lerp(x1, x2, i/40.0) + 40;

float y = lerp(y1, y2, i/40.0);

point(x, y);

}

// map() Rimappa un numero da un intervallo a un altro

Codici dei tasti CODED, reperibili con keyCode:

UP, DOWN, LEFT, RIGHT, ALT, CONTROL e SHIFT

Mentre in key, non avendo bisogno di fare il controllo se key==CODED, abbiamo i seguenti caratteri speciali:

BACKSPACE, TAB, ENTER, RETURN, ESC e DELETE.

L'invio di può "catturare" in due modi (key == ENTER) oppure (key == '\n').

Lo spazio è il carattere ' ' (c'è uno spazio tra i singoli apici)

**Lezione 4**

textSize(32); //dimensione testo

text("word",10,30);

fill(0,100,150);

text("word",10,60); //testo

fill(0,100,150,50);

text("word",10,90);

void setup(){

size(120,120);

//quanto sarà grande e allineamento testo

textSize(64);

textAlign(CENTER);

}

void draw (){

background(0);

text(key, 60,80); //key è una variabile CHAR che contiene il carattere del tasto premuto

}

void setup(){

size(120,120);

//quanto sarà grande e allineamento testo

textSize(64);

textAlign(CENTER);

}

void draw (){

background(0);

text(key, 60,80); //key è una variabile CHAR che contiene il carattere del tasto premuto

}

//si possono definire stringe con string

void setup() {

size(250, 250);

}

void draw() {

background(0);

translate(width/2, height/2); // Move coordinate system to center of sketch

rotate(millis() \* 0.001 \* TWO\_PI / 10.0); // Move 360 degrees in ten second

translate(80, 0); // Move away from the center

rotate(millis() \* 0.001 \* TWO\_PI); // One revolution every second

/\* Ellipse \*/

fill(255);

noStroke();

ellipseMode(CENTER);

ellipse(0, 0, 40, 40);

/\* Line \*/

strokeWeight(5);

stroke(255, 0, 0);

line(0, 0, 40, 0);

}

int x=0;

void setup(){

size(500,500);

rectMode(CENTER);

}

void draw(){

//background(255);

pushMatrix();

translate(x+300,200);

rotate(frameCount%360);

fill(0,255,0);

rect(0,0,40,40);

popMatrix();

pushMatrix();

translate(x+100,200);

rotate(frameCount%360);

fill(0,0,255);

rect(0,0,40,40);

popMatrix();

x++;

}

**Lezione 5**

PImage im; // variabile per immagine

void setup(){

size(600,600);

im=loadImage("lena.png"); //loading di un immagine

println("Larghezza: "+im.width);

println("Altezza: "+im.height);

}

void draw(){

background(0);

image(im,0,0);

}

void mousePressed(){

im.resize(im.width/2,im.height/2); //ridimensionare un immagine

}

void keyTyped(){

if (key=='s' || key=='S'){

println(savePath("lenaPiccola.png"));

im.save(savePath("lenaPiccola.png"));

}

if (key=='c' || key=='C'){

println("Salvo la canvas...");

saveFrame("lenaCanvas.png");

}

}

//Si possono salvare immagini tramite url

Filter()

Filtra l'immagine come definito da una delle seguenti modalità:

SOGLIA

Converte l'immagine in pixel in bianco e nero a seconda che si trovino sopra o al di sotto della soglia definita dal parametro level. Il parametro deve essere tra 0,0 (nero) e 1,0 (bianco). Se non viene specificato alcun livello, 0,5 è usato.

GRIGIO

Converte tutti i colori dell'immagine in equivalenti in scala di grigi. Nessun parametro è usato.

OPACO

Imposta il canale alfa su completamente opaco. Non viene utilizzato alcun parametro.

INVERSIONE

Imposta ogni pixel sul relativo valore inverso. Non viene utilizzato alcun parametro.

POSTERIZE

Limita ogni canale dell'immagine al numero di colori specificati come parametro. Il parametro può essere impostato su valori compresi tra 2 e 255, ma i risultati sono più evidenti nelle gamme inferiori.

BLUR

Esegue una sfocatura gaussiana con il parametro level che specifica l'estensione di la sfocatura. Se non viene utilizzato alcun parametro, la sfocatura equivale a Gaussian Sfocatura del raggio 1. Valori più grandi aumentano la sfocatura.

ERODE

Riduce le zone luminose. Non viene utilizzato alcun parametro.

DILATARE

Aumenta le aree di luce. Non viene utilizzato alcun parametro.

THRESHOLD

GRAY

OPAQUE

INVERT

POSTERIZE

BLUR

ERODE

DILATE

**Lezione 6**

Quantizzazione, rotazione, scaling, interpolazione

**Lezione 7**

Contrasto, contrasto istogramma, op\_puntuali, op\_puntuali\_mouse

**Lezione 8**

Convoluzione, rango, bitplane