



PPEM 2017

Imágenes, parte 1

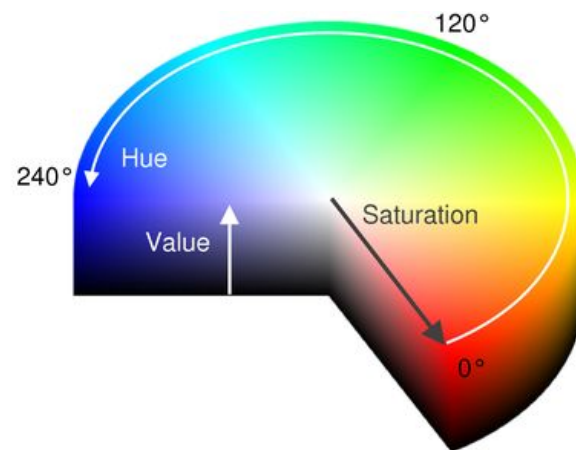
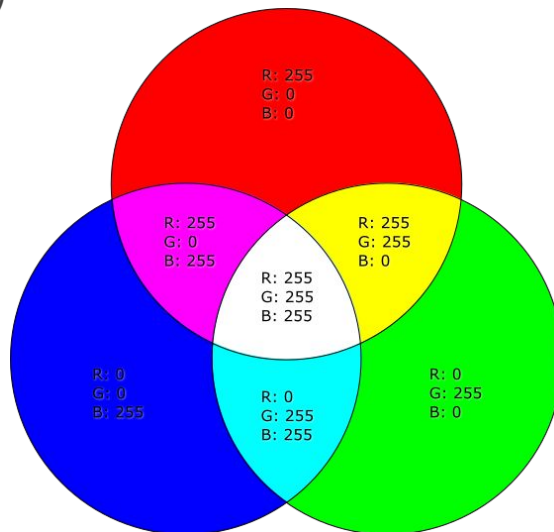
Repaso

Figuras (arc, triangle, vertex)

Espacios de colores

Mouse y teclado

Librerías externas



```
shape(shape)
shape(shape, x, y)
shape(shape, x, y, w, h)
```

Cargar imagen svg como PShape

PShape s;

Carga svg (creados en Inkscape and Adobe Illustrator)
o obj (modelos 3D)

```
void setup() {
  size(500, 500);
  s = loadShape("hs.svg");
}
```

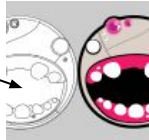
```
void draw() {
  shape(s, 10, 10, width-10, height-10);
}
```

PShape

isVisible()

setVisible()

disableStyle()



enableStyle()

scale()

+

shapeMode: CENTER o CORNER

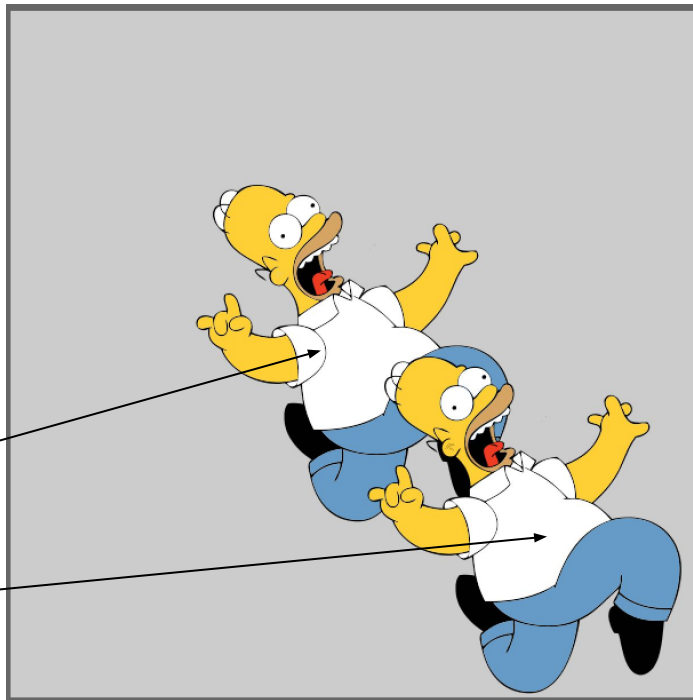
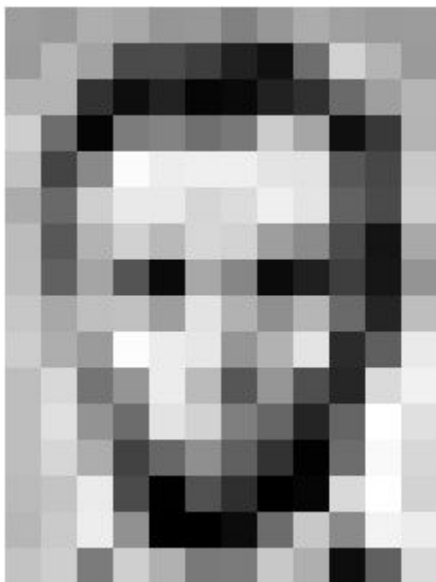


Imagen como array de píxeles

Array de pixeles



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	6	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	35	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

How the pixels look:

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

How the pixels are stored:

0	1	2	3	4	5	6	7	8	9	.	.	.		
---	---	---	---	---	---	---	---	---	---	---	---	---	--	--



Para primeros ejercicios con las imágenes

1. Guardamos un sketch “fotoTest”
2. En la carpeta “fotoTest” generada en el espacio de trabajo de Processing creamos una carpeta “data”
3. Bajamos una imagen cualquiera de internet (formato .gif, .jpg, .tga, .png) y guardamos en la carpeta “data”
4. Escribimos siguiente código:

```
loadImage(filename)  
loadImage(filename, extension)
```

Cargar una imagen de la computadora

```
PImage localImg;
```

```
void settings() {
```

```
  localImg = loadImage("pic.jpg"); // path local si está en el folder "data" adentro del folder del sketch  
  // localImg = loadImage("/Users/ewe/dev/p3/sketch_170807a/data/pic.jpg"); path absoluto si no está  
  size(localImg.width, localImg.height);
```

```
}
```

```
void draw() {
```

```
  background(0);  
  image(localImg, 0, 0); // función para "mostrar" la imagen
```

```
}
```

```
loadImage(filename)  
loadImage(filename, extension)
```

Cargar una imagen de la web

```
PImage webImg;
```

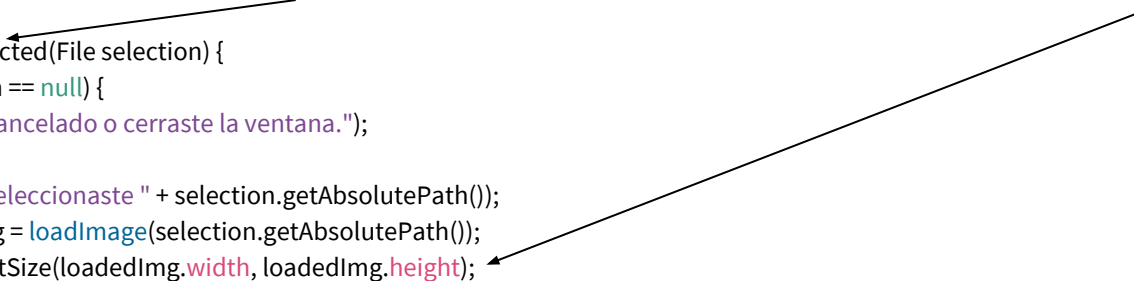
```
void settings() {  
  String url = "https://processing.org/img/processing-web.png";  
  webImg = loadImage(url, "png");  
  size(webImg.width, webImg.height);  
}
```

```
void setup() {  
  background(0);  
  image(webImg, 0, 0);  
}
```




Cargar una imagen en el momento

```
PIImage loadedImg;
void settings() {
    loadedImg = loadImage("we.jpg");
    size(loadedImg.width, loadedImg.height);
}
void draw(){
    image(loadedImg,0,0);
}
void mouseClicked(){
    selectInput("Selecciona imagen:", "fileSelected");
}
void fileSelected(File selection) {
    if (selection == null) {
        println("Cancelado o cerraste la ventana.");
    } else {
        println("Seleccionaste " + selection.getAbsolutePath());
        loadedImg = loadImage(selection.getAbsolutePath());
        surface.setSize(loadedImg.width, loadedImg.height);
    }
}
```



Image

`image(img, x, y)` // (x,y) de la esquina superior izquierda, ancho y alto de la imagen
`image(img, x, y, w, h)` // forzamos el w y h

```
PImage img;  
  
void setup() {  
  size(200, 200);  
  img = loadImage("we.jpg");  
}  
  
void draw() {  
  background(255);  
  image(img, 10, 10, 70, 140);  
}
```

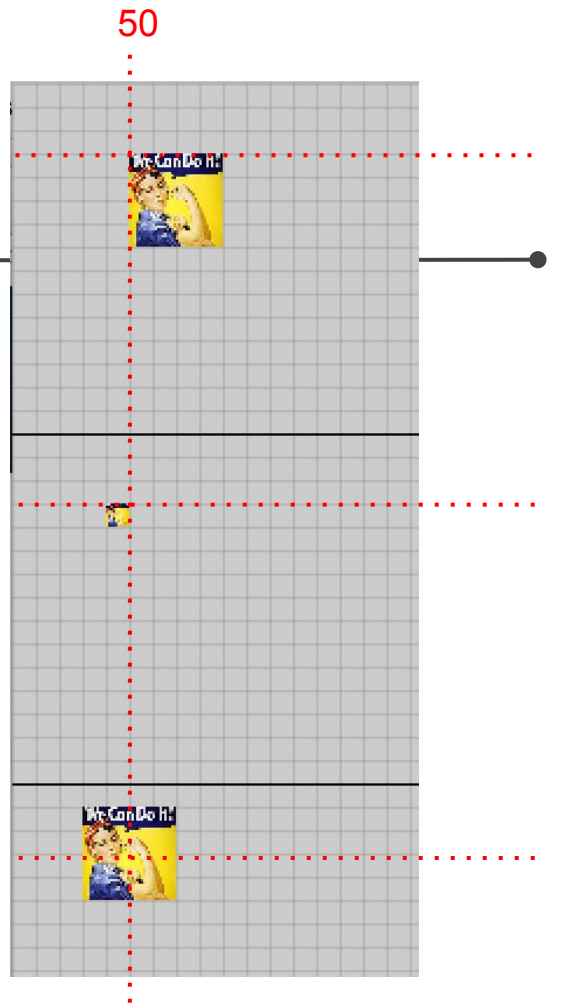


imageMode

```
PImage img = loadImage("we.jpg");  
size(400,800);  
imageMode(CORNER);  
image(img, 50, 30, 40, 40); // image(img,x,y,w,h), (x,y) de la esquina superior izquierda
```

```
translate(0,150);  
line(0,0,width,0);  
imageMode(CORNERS);  
image(img, 50, 30, 40, 40); // image(img,x,y,x2,y2)
```

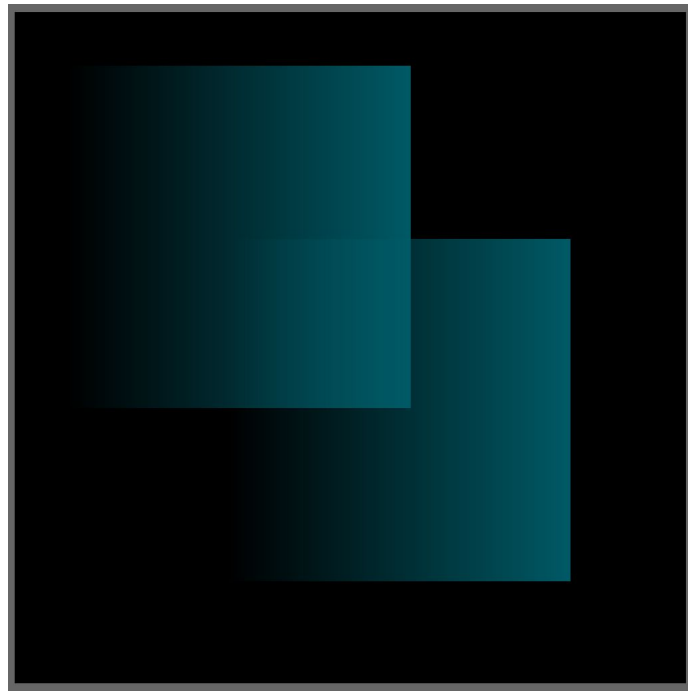
```
translate(0,150);  
line(0,0,width,0);  
imageMode(CENTER);  
image(img, 50, 30, 40, 40); // image(img,x,y, w, h), (x,y) del centro
```



Crear una imagen // array pixels[]

```
PIimage img;  
void setup(){  
  size(500,500);  
  img = createImage(255, 255, ARGB);  
  for (int i = 0; i < img.pixels.length; i++) {  
    img.pixels[i] = color(0, 90, 102, i % img.width);  
  }  
}  
  
void draw(){  
  background(0);  
  image(img, 40, 40);  
  image(img, mouseX-img.width/2, mouseY-img.height/2);  
}
```

Seteo el color pixel por pixel



```
pimg.set(x, y, c)
pimg.set(x, y, img)
```

Crear una imagen // img.set(x,y,c)

```
PImage img;
void setup(){
  size(500,500);
  img = createImage(255, 255, ARGB);
  for (int i = 0; i < img.width; i++) {
    for (int j = 0; j < img.height; j++) {
      img.set(i,j,color(0, 90, 102, i));
    }
  }
}
```

Seteo el color pixel por pixel

```
void draw(){
  background(0);
  image(img, 40, 40);
  image(img, mouseX-img.width/2, mouseY-img.height/2);
}
```

```
pimg.set(x, y, c)
pimg.set(x, y, img)
```

Combinación de imágenes con set()

```
PImage img, img2;
```

```
size(350,350);
```

```
background(0);
```

```
img = loadImage(255, 255, ARGB);
```

```
img2 = loadImage("we.jpg");
```

```
for (int i = 0; i < img.width; i++) {
```

```
  for (int j = 0; j < img.height; j++) {
```

```
    img.set(i,j,color(0, 90, 102, i));
```

```
  }
```

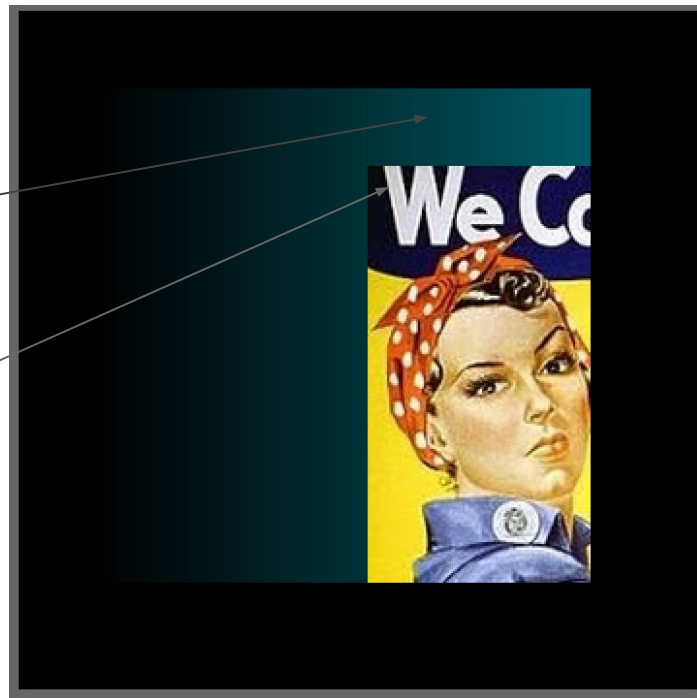
```
}
```

```
img.set(140,40,img2);
```

```
image(img, 40, 40);
```

Seteo el color pixel por pixel

Seteo los pixeles de una imagen



```
vertex(x, y, u, v)
```

Imagen como textura

```
PImage img;
```

```
void setup() {  
  size(400, 400, P3D);  
  img = loadImage("we.jpg");  
}
```

```
void draw() {  
  background(255);  
  beginShape();  
  texture(img);  
  vertex(20, 40, 0, 0);  
  vertex(width, 0, img.width, 0);  
  vertex(width-40, height-20, img.width, img.height);  
  vertex(0, height-40, 0, img.height);  
  endShape();  
}
```



```
ping.resize(w, h)
```

Resize

```
size(400,400);
```

```
PImage img = loadImage("we.jpg");
```

```
image(img,10,10,20,60);
```

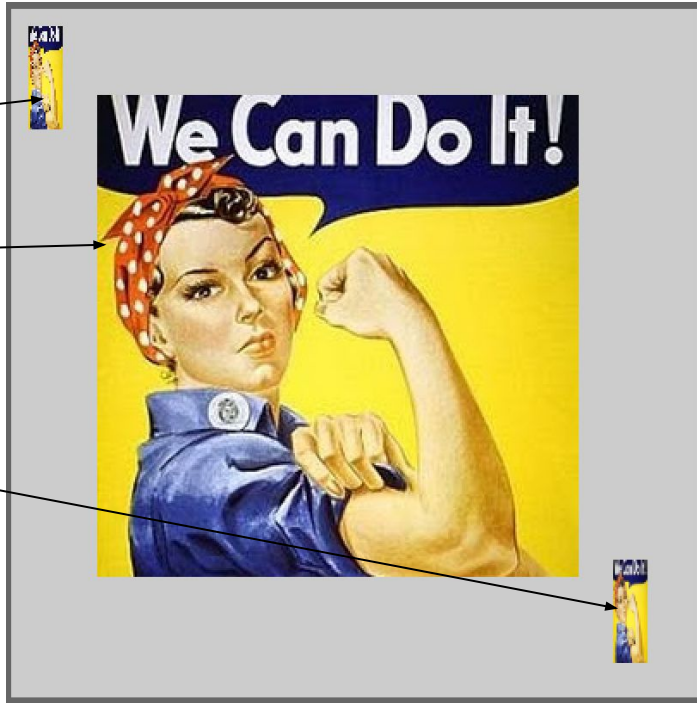
```
image(img, 50, 50);
```

```
img.resize(20,60);
```

```
image(img, 350, 320);
```

! cambia las dimensiones de la imagen !

! si uno de los parámetros == 0, se escala en proporción con el otro !




```
pygame.get(x, y)
pygame.get(x, y, w, h)
pygame.get()
```

Get

Para consultar colores de los pixeles, conseguir un “recorte” o copia de una imagen

```
size(400,500);
PImage img = loadImage("we.jpg");
```

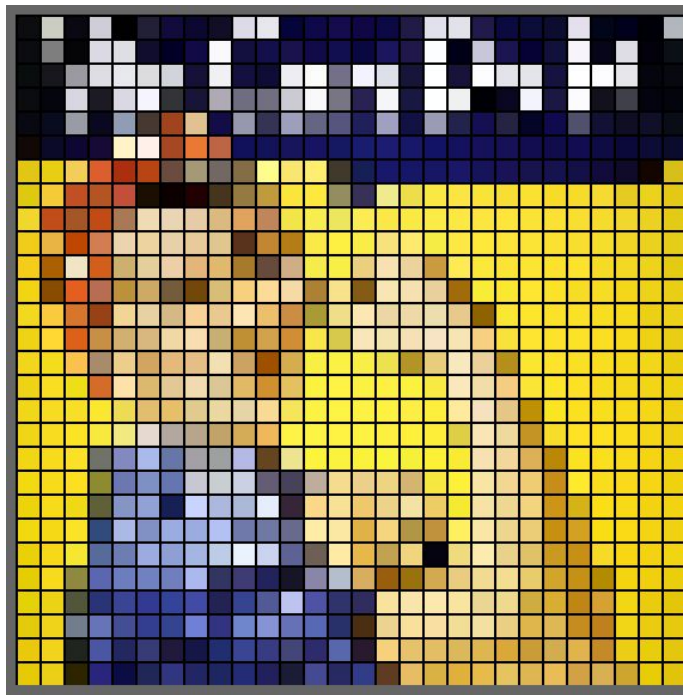
```
image(img, 50, 20);
color c = img.get(img.width-1, img.height/2);
fill(c);
rect(40, 10, 20, 20);
```

```
PImage part = img.get(10, 100, img.width/2, img.height/2);
image(part, 50, 320);
```



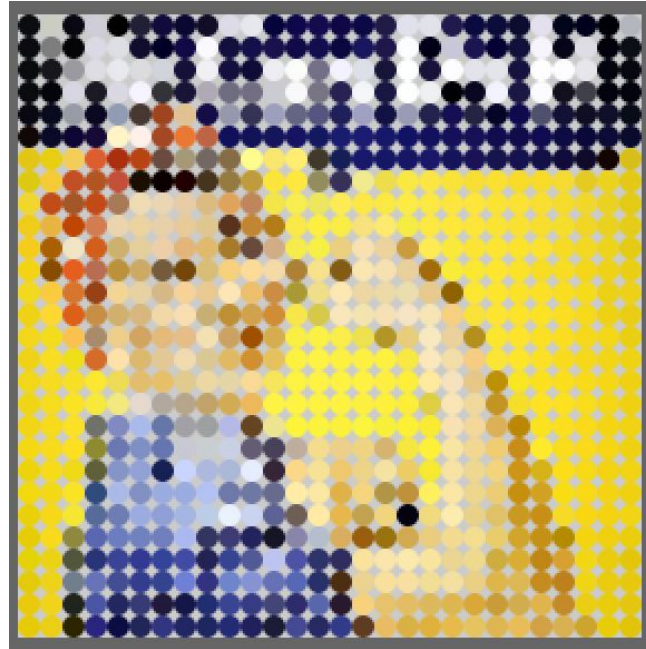
Get y imagenes pixeladas

```
PImage img;  
void settings() {  
  img = loadImage("we.jpg");  
  size(img.width, img.height);  
}  
void setup(){  
  for (int i = 0; i < img.width; i+=10) {  
    for(int j = 0; j < img.height; j+=10){  
      fill(img.get(i,j));  
      rect(i,j,10,10);  
    }  
  }  
}
```



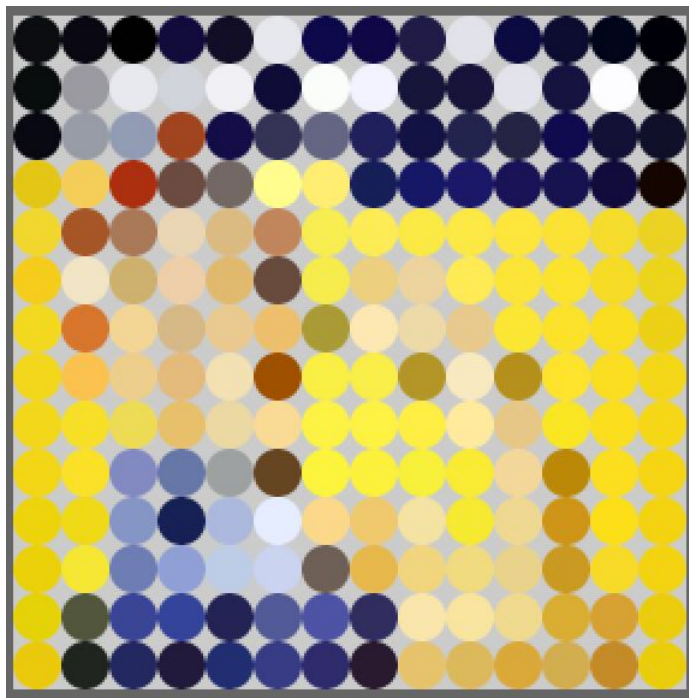
Get y pixeles circulares

```
PImage img ;  
void settings() {  
  img = loadImage("we.jpg");  
  size(img.width, img.height);  
}  
void setup(){  
  noStroke();  
  ellipseMode(CORNER);  
  for (int i = 0; i < img.width; i+=10) {  
    for (int j = 0; j < img.height; j+=10){  
      fill(img.get(i,j));  
      ellipse(i,j,10,10);  
    }  
  }  
}
```



Más abstracto?

```
PImage img ;
int figureWH = 20;
void settings() {
  img = loadImage("we.jpg");
  size(img.width, img.height);
}
void setup(){
  noStroke();
  ellipseMode(CORNER);
  for (int i = 0; i < img.width; i+=figureWH) {
    for (int j = 0; j < img.height; j+=figureWH){
      fill(img.get(i,j));
      ellipse(i,j,figureWH,figureWH);
    }
  }
}
```



Modificar píxel por píxel

Bajamos de internet dos imágenes cualquiera de la misma resolución y cambiamos cada segundo píxel de la imagen 1 a la imagen 2.



Intercambio de píxeles

```
size(400,500);  
PImage img = loadImage("we.jpg");  
PImage img2 = loadImage("hamster.jpg");  
  
for (int i = 0; i < img.width; i+=2) {  
  for (int j = 0; j < img.height; j++){  
    img.set(i,j, img2.get(i,j));  
  }  
}
```

image(img, 0, 0);



Cambio de píxeles

```
int col = 0;
PImage img, img2;
void setup(){
  size(280,280);
  img = loadImage("we.jpg");
  img2 = loadImage("hamster.jpg");
  frameRate(30);
}
void draw(){
  for(int j = 0; j < img.height; j++){
    img.set(col, j, color(img2.get(col, j), 10));
  }
  col += 1;
  image(img, 0, 0);
}
```



```
pimg.copy()  
pimg.copy(sx, sy, sw, sh, dx, dy, dw, dh)  
pimg.copy(src, sx, sy, sw, sh, dx, dy, dw, dh)
```

Copy = get, set y resize en uno

```
PImage img, img2, img3;  
void setup(){  
  size(280,280);  
  img = loadImage("we.jpg");  
  img2 = loadImage("hamster.jpg");  
  img3 = createImage(img.width, img.height, RGB);  
}  
void draw(){  
  int randomX = ceil(random(img.width - 10));  
  int randomY = ceil(random(img.height - 10));  
  if(frameCount%2 == 0)  
    img3.copy(img, randomX, randomY, 10, 10, randomX, randomY, 10, 10);  
  else  
    img3.copy(img2, randomX, randomY, 10, 10, randomX, randomY, 10, 10);  
  image(img3, 0, 0);  
}
```



Tint

```
tint(rgb)
tint(rgb, alpha)
tint(gray)
tint(gray, alpha)
tint(v1, v2, v3)
tint(v1, v2, v3, alpha)
```

```
colorMode(HSB);
size(500,500);
PImage img = loadImage("we.jpg");
background(255);
image(img,0,0);
for(int i = 10; i<width; i+=50){
  translate(50,50);
  tint(random(255),255,255);
  image(img,0,0);
}
```



Tint con transparencia

```
tint(rgb)
tint(rgb, alpha)
tint(gray)
tint(gray, alpha)
tint(v1, v2, v3)
tint(v1, v2, v3, alpha)
```

tint(rgb)

tint(rgb, alpha)

tint(gray)

tint(gray, alpha)

tint(v1, v2, v3)

tint(v1, v2, v3, alpha)

