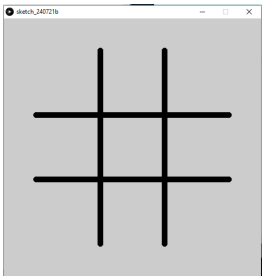
Processing. Juegos. Gato (*tic tac toe*, tres en línea, etcétera)

Miguel Navarro Saad

Quiero explicar cómo uso `PGraphics`. Primero declaro `uno` y `dos` y los creo en **`setup`**, donde también defino `size`(500, 500) al principio.

```
int delta = 5;
int lado;

PGraphics uno, dos;

void setup() {
  size(500, 500);
  lado = width/4-2*delta;
  uno = createGraphics(lado, lado);
  dos = createGraphics(lado, lado);
}

void draw() {
  gato();
}
```

Al final del programa diseño el gato,



```
void gato() {
  stroke(0);
  strokeWeight(2*delta);
  line(3*width/8, height/8, 3*width/8, 7*height/8);
  line(5*width/8, height/8, 5*width/8, 7*height/8);
  line(width/8, 3*height/8, 7*width/8, 3*height/8);
  line(width/8, 5*height/8, 7*width/8, 5*height/8);
}
```

y en seguida la cruz y el círculo

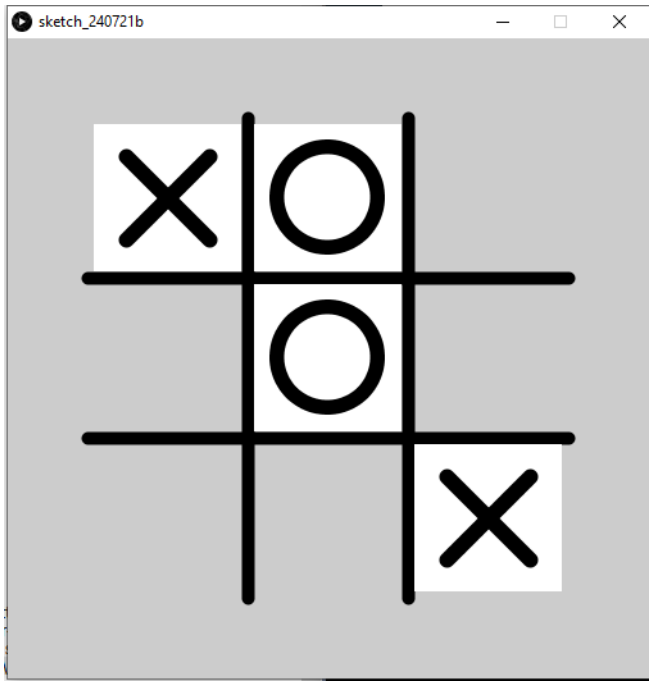
```
void cruz() {
  uno.beginDraw();
  uno.background(255, 255, 255);
  uno.stroke(0, 0, 0);
  uno.strokeWeight(diametro/8);
```

```
  uno.line(diametro/4, diametro/4, lado-diametro/4, lado-diametro/4);
  uno.line(lado-diametro/4, diametro/4, diametro/4, lado-diametro/4);
  uno.endDraw();
}

void circulo() {
  dos.beginDraw();
  dos.background(255, 255, 255);
  dos.noStroke();
  dos.fill(0, 0, 0);
  dos.circle(lado/2, lado/2, lado-diametro/4);
  dos.fill(255, 255, 255);
  dos.circle(lado/2, lado/2, 3*(lado-diametro/4)/4);
  dos.endDraw();
}
```

La cruz la guardo en el gráfico `uno` y el círculo en el gráfico `dos`. Ahora puedo poner estos dos gráficos donde quiera las veces que quiera utilizando `image` desde **draw**. Sólo que para no llamarlos cada vez que se ejecuta **draw** los llamo una única vez desde **setup**

```
int delta = 5;
int lado;
int diametro = 100;

PGraphics uno, dos;

void setup() {
  size(500, 500);
  lado = width/4-2*delta;
  uno = createGraphics(lado, lado);
  dos = createGraphics(lado, lado);
  gato();
  cruz();
  circulo();
}

void draw() {
  image(uno, width/8+delta, height/8+delta);
  image(dos, 3*width/8+delta, height/8+delta);
  image(uno, 5*width/8+delta, 5*height/8+delta);
  image(dos, 3*width/8+delta, 3*height/8+delta);
}

void gato() {
  stroke(0);
  strokeWeight(2*delta);
  line(3*width/8, height/8, 3*width/8, 7*height/8);
  line(5*width/8, height/8, 5*width/8, 7*height/8);
  line(width/8, 3*height/8, 7*width/8, 3*height/8);
```

```
  line(width/8, 5*height/8, 7*width/8, 5*height/8);
}

void cruz() {
  uno.beginDraw();
  uno.background(255, 255, 255);
  uno.stroke(0, 0, 0);
  uno.strokeWeight(diametro/8);
  uno.line(diametro/4, diametro/4, lado-diametro/4, lado-diametro/4);
  uno.line(lado-diametro/4, diametro/4, diametro/4, lado-diametro/4);
  uno.endDraw();
}

void circulo() {
  dos.beginDraw();
  dos.background(255, 255, 255);
  dos.noStroke();
  dos.fill(0, 0, 0);
  dos.circle(lado/2, lado/2, lado-diametro/4);
  dos.fill(255, 255, 255);
  dos.circle(lado/2, lado/2, 3*(lado-diametro/4)/4);
  dos.endDraw();
}
```



Cambiamos el fondo en **setup** para corregirlo

```
background(255, 255, 255);
```

De acuerdo, eso es en cuanto a generar los gráficos y una muestra de cómo usarlos. Pero para hacer fubcionar el juego tenemos que leer dónde está el cursor, dónde poner uno u otro gráfico dependiendo del turno de cada jugador y bloquear el acceso a los lugares ocupados; adicionalmente, debemos saber cuándo termina el juego e indicarlo.

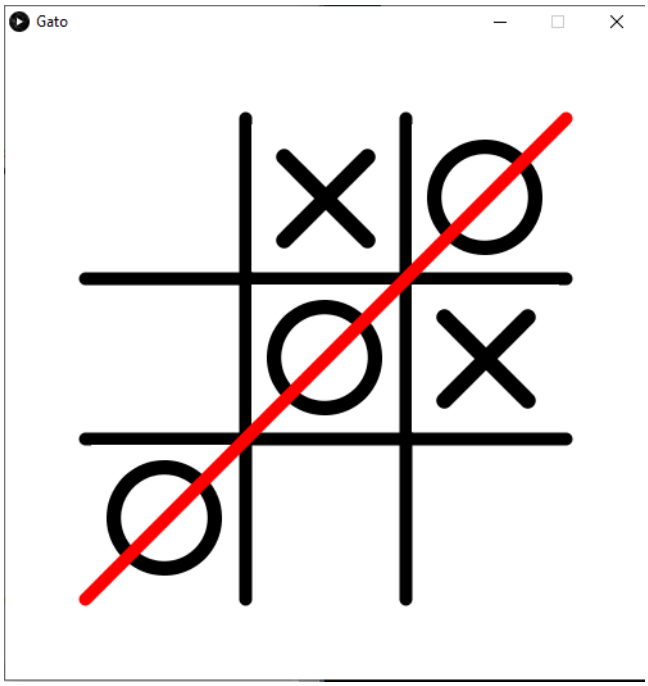
Processing es muy flexible y por ello para saber la posición del cursor y dónde poner los gráficos, declaramos una matriz de 3×3 (las nueve casillas del gato) de vectores (x y y iniciales y finales de los gráficos) que creamos en **setup**

```
pos = new PVector[4][4];
for (int i = 0; i <= 3; i++)
  for (int j = 0; j <= 3; j++)
    pos[i][j] = new PVector((2*j+1)*width/8+delta, (2*i+1)*height/8+delta);
```

Para marcar las casillas ocupadas y para saber cuándo termina el juego utilizamos una matriz de enteros: cero (0) significa que está disponible, uno (1) que está ocupada por una cruz y dos (2) que está ocupada por un círculo. Así, **draw** queda como sigue

```
void draw() {
  mX = mouseX;
  mY = mouseY;
  if (mousePressed) {
    for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++)
        if (mX > pos[i][j].x && mX < pos[i][j+1].x &&
            mY > pos[i][j].y && mY < pos[i+1][j].y)
          if (estado[i][j] == 0) {
            cambia = !cambia;
            estado[i][j] = (cambia)?1:2;
            image((cambia)?uno:dos, pos[i][j].x, pos[i][j].y);
          }
  }
  fin();
}
```

donde `fin` revisa columnas, renglones y diagonales para saber cuando termina el juego.



Desafortunadamente no conozco la manera de programar un reinicio con *Processing*, pero si se crea la `app` para *Android* basta con girar el móvil para reiniciar.

Podríamos crear figuras personales usando `beginShape`, `vertex` y `endShape`... Tal vez crear un rompecabezas o un tablero de ajedrez con sus 32 gráficos para las piezas.

En mi programa `RellenaLetra.pde` pinto sobre el lienzo y sobre éste pongo un gráfico (dejando al descubierto en el lienzo el selector de colores) con las letras del alfabeto 'perforadas', dando la impresión de que coloreo la letra.