Unidad 23

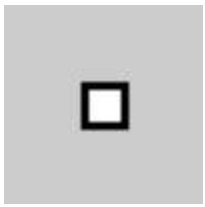
Valores de Entrada: Teclado

Elementos que se introducen en esta Unidad:
`keyPressed, key, keyCode`

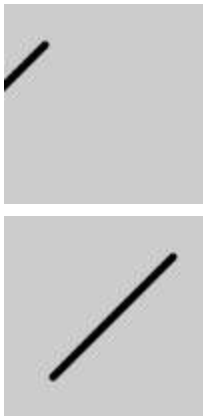
Los teclados son comúnmente utilizados como dispositivos de entrada de caracteres para componer documentos de texto, Email, mensajería instantánea, y relacionados. Sin embargo, el potencial de un teclado de computadora va mucho más allá que su uso común. La expansión de la máquina de escribir al teclado permitió utilizar este mismo como medio para ejecutar programas, moverse entre aplicaciones y navegar en 3D en los diversos video-juegos. Es posible ignorar la impresión de caracteres y concentrarse solo en el ritmo en el que se oprimen las teclas. De esta forma, podremos crear un programa que obtenga como valor de entrada la velocidad con la que se escribe, y utilizar dichos valores para controlar la velocidad de un evento. En los últimos años, el teclado ha sido revalorizado por otra clase de dispositivos que no responden al estilo de escritura QWERTY. Métodos de tamaño más pequeño, como el teclado numérico de los teléfonos (a pesar de que dicha interfaz halla fallado en la industria y ahora los teléfonos móviles utilicen teclado tipo QWERTY). Por otro lado, el reconocimiento de voz está avanzando fuertemente, y se proclama como una alternativa futura a la escritura por medio del teclado.

-Datos del Teclado

Processing puede registrar la última tecla presionada por un dispositivo de teclado. Por lo tanto, la variable que se emplea es denominada `keyPressed`, y devuelve un valor de tipo `boolean`. Dicho valor será `true` solo cuando una tecla halla sido presionada, de lo contrario será `false`. Mientras la tecla es presionada, este valor se mantendrá `true`, sin embargo, se convertirá en un `false` cuando la tecla sea liberada.

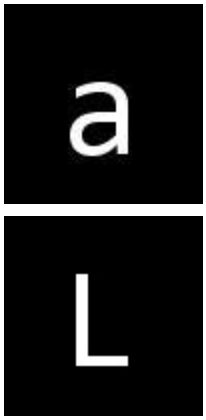


```
// Dibuja una línea si alguna tecla está siendo presionada
void setup() {
      size(100, 100);
      smooth();
      strokeWeight(4);
}
void draw() {
      background(204);
      if (keyPressed == true) {     // Si una telcla es
            line(20, 20, 80, 80);   // presionada, dibuja una
      } else {                      // línea. Sino,
            rect(40, 40, 20, 20);   // dibuja un rectángulo
      }
}
```



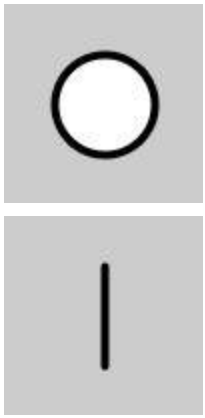
```
// Mueve una línea mientras una tecla es presionada
int x = 20;
void setup() {
      size(100, 100);
      smooth();
      strokeWeight(4);
}
void draw() {
      background(204);
      if (keyPressed == true){  // Si una tecla es presionada
            x++;                // agrega 1 a x
      }
      line(x, 20, x-60, 80);
}
```

La variable `key` es del tipo `char`, permite almacenar un carácter, el último presionado. La variable `key` solo almacenará un valor cada vez. De esta manera, podemos crear un programa que muestre texto utilizando la variable `key` y la función `text()`.




```
PFont font;
void setup() {
      size(100, 100);
      fuente = loadFont("ThesisMonoLight-72.vlw");
      textFont(fuente);
}
void draw() {
      background(0);
      text(key, 28, 75);
}
```

La variable `key`, suele emplearse para detectar cual fue la última tecla oprimida. El siguiente ejemplo usa la expresión relacional `key == 'A'`. El empleo de las comillas simples (`' '`) y no de las comillas dobles (`" "`) se debe a que las comillas doble se interpretan como valores del tipo `String`, y `key` solo devuelve valores del tipo `char`. Por lo tanto, si utilizáramos comillas dobles nos produciría un error. El evento se producirá cuando se oprima la A mayúscula únicamente.

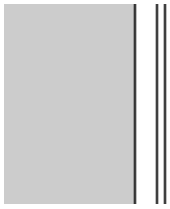


```
void setup() {
      size(100, 100);
      smooth();
      strokeWeight(4);
}
void draw() {
      background(204);
      // Si 'A' es presionada, dibuja una línea
      if ((keyPressed == true) && (key == 'A')) {
            line(50, 25, 50, 75);
      } else {          // Sino, dibuja una elipse
            ellipse(50, 50, 50, 50);
      }
}
```

En contadas ocasiones necesitaremos detectar si una tecla, que producirá un evento, es presionada, pero se puede correr el riesgo de que el usuario tenga activada la mayúscula. Por ende, vamos a recurrir a comprobar ambas teclas para un mismo evento. Para esto, utilizaremos la expresión relacional `||` (o).

```
if ((keyPressed == true) && ((key == 'a') || (key == 'A'))) {
```

Ya que cada tecla tiene un valor numérico asignado (ASCII), el valor de una `key` puede utilizarse como variable de control.



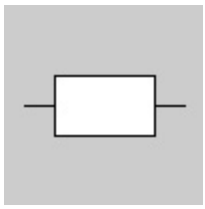
```
int x = 0;
void setup() {
      size(100, 100);
}
void draw() {
      if (keyPressed == true) {
```

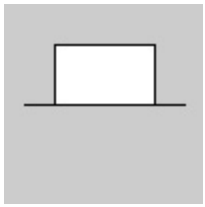
```
            x = key - 32;
            rect(x, -1, 20, 101);
      }
}
```

```
float angle = 0;
void setup() {
      size(100, 100);
      smooth();
      strokeWeight(8);
}
void draw() {
      background(204);
      if (keyPressed == true) {
            if ((key >= 32) && (key <= 126)) {
                  // Si es una tecla alfanumérica,
                  // convertir este valor en un ángulo
                  angle = map(key, 32, 126, 0, TWO_PI);
            }
      }
      arc(50, 50, 66, 66, angle-PI/6, angle+PI/6);
}
```

-Teclas Codificadas

Además de leer letras numéricas, caracteres, y símbolos, Processing puede leer valores de otras letras tales como Alt, Control, Shift, Barra-Espaciadora, Enter, Return, Escape y la barra de borrar. La variable `keyCode` almacena constantes encargadas de administrar esa información. Se emplea con la expresión `keyCode == CODED`, donde la constante puede ser `ALT`, `CONTROL`, `SHIFT`, `UP` (arriba), `DOWN` (abajo), `LEFT` (izquierda), y `RIGHT` (derecha). Existen caracteres que no son alfanuméricos, como `BACKSPACE`, `TAB`, `ENTER`, `RETURN`, `ESC`, y `DELETE`, y sin embargo no se los considera caracteres codificados. Esto se debe a que su implementación es muy variable dependiendo la plataforma en la que se realicen. Se verá más adelante.





```
int y = 35;
void setup() {
      size(100, 100);
}
void draw() {
      background(204);
      line(10, 50, 90, 50);
      if (key == CODED) {
            if (keyCode == UP) {
                  y = 20;
            } else if (keyCode == DOWN) {
                  y = 50;
            }
      } else {
            y = 35;
      }
      rect(25, y, 50, 30);
}
```