Unidad 21

Valores de Entrada: Mouse

Elementos que se introducen en esta Unidad:
`mouseX, mouseY, pmouseX, pmouseY, mousePressed, mouseButton`
`cursor(), noCursor()`

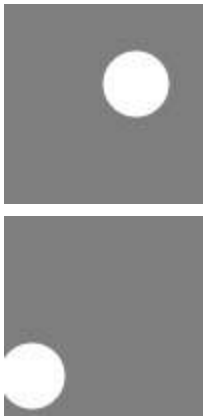
La pantalla del ordenador es tan solo un puente entre nuestro cuerpo físico y una cantidad abismal de circuitos eléctricos que habitan dentro de una computadora. Controlamos los elementos de la pantalla a través de "prótesis" físicas, tales como pantallas táctiles, trackballs o joysticks. Sin embargo, el más común de todos estos dispositivos posiblemente sea el *mouse* (ratón). El mouse de los ordenadores data de finales de 1960, cuando Douglas Engelbart lo presentó como un dispositivo del oN-Line System (NLS), uno de los primeros sistemas de ordenadores con visualización de video. El mouse fue incluido como concepto en Xerox Palo Alto Research Center (PARC), pero no fue hasta 1984 que la Apple Macintosh lo convirtió en catalizador de su actual uso. El diseño del mouse ha tenido diversas modificaciones a través de los años. No obstante, su forma de uso sigue siendo la mismo. Se encarga de cambiar en pantalla la posición X-Y del cursor.

-Datos del Mouse

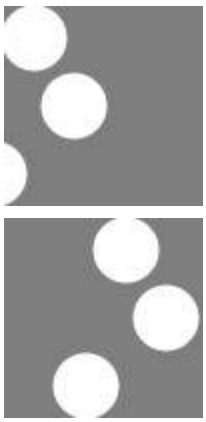
En Processing, las variables que nos permiten obtener datos del mouse son `mouseX` y `mouseY` (nótese el uso de mayúsculas en la X y en la Y) tomando como referencia la esquina superior izquierda como eje 0,0. Para ver los valores actuales del mouse en la consola, se recurre a un simple programa:

```
void draw() {
      frameRate(12);
      println(mouseX + " : " + mouseY);
}
```

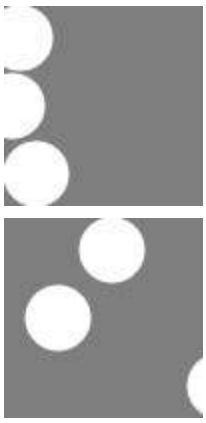
Cuando un programa inicia, el valor de `mouseX` y `mouseY` es 0. Si el cursor se mueve dentro de la ventana de representación, el valor cambia a la actual posición del mismo. Si el cursor se encuentra a la izquierda, el valor de `mouseX` será 0 y comenzará a incrementar a medida que este se mueve hacia la derecha. En cambio, si el cursor esta arriba, el valor de `mouseY` será 0 y comenzará a aumentar a medida que este se mueve hacia abajo. Si `mouseX` y `mouseY` se encuentran en un programa donde no existe una estructura `draw()` o está activada la función `noLoop()` en el `setup()`, los valores de ambos serán siempre 0. Generalmente, la posición del mouse es utilizada para controlar la posición de algunos elementos de pantalla. Lo interesante se produce cuando existen diferentes relaciones entre unos elementos y otros a base de conseguir datos de entrada por la posición del mouse. Para invertir los valores del mouse, simplemente hay que restarle a `mouseX` el ancho de pantalla (`width`) y a `mouseY` el alto de pantalla (`height`).



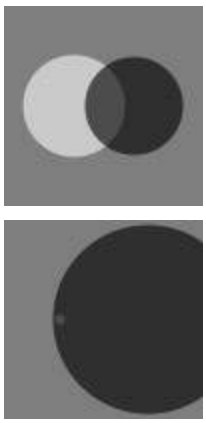
```
// Un circulo sigue al cursor
void setup() {
      size(100, 100);
      smooth();
      noStroke();
}
void draw() {
      background(126);
      ellipse(mouseX, mouseY, 33, 33);
}
```



```
//Agregando operaciones
void setup() {
      size(100, 100);
      smooth();
      noStroke();
}
void draw() {
      background(126);
      ellipse(mouseX, 16, 33, 33);        //Circulo de Arriba
      ellipse(mouseX+20, 50, 33, 33);     //Circulo de el Medio
      ellipse(mouseX-20, 84, 33, 33);     //Circulo de Abajo
}
```



```
//Al multiplicar y dividir se crean posiciones escalares
void setup() {
      size(100, 100);
      smooth();
      noStroke();
}
void draw() {
      background(126);
      ellipse(mouseX, 16, 33, 33);        //Circulo de Arriba
      ellipse(mouseX/2, 50, 33, 33);      //Circulo de el Medio
      ellipse(mouseX*2, 84, 33, 33);      //Circulo de Abajo
}
```



```
//Invertir la posición del cursor para crear segundas
//respuestas
void setup() {
      size(100, 100);
      noStroke();
      smooth();
}
void draw() {
      float x = mouseX;
      float y = mouseY;
      float ix = width - mouseX;          //Invertir X
      float iy = mouseY - height;         //Invertir Y
      background(126);
      fill(255, 150);
      ellipse(x, height/2, y, y);
      fill(0, 159);
      ellipse(ix, height/2, iy, iy);
}
```

Las variables de Processing, `pmouseX` y `pmouseY` imprimen como valor la posición del mouse previa al cuadro que se esta ejecutando. Si el mouse no se mueve, el valor siempre será el mismo. Sin embargo, si el mouse se mueve rápidamente, lo valores pueden oscilar entre diversos parámetros. Para ver esta diferencia, se puede ejecutar un simple programa que alterne los movimientos lentos y rápidos del mouse:

```
void draw() {
      frameRate(12);
      println(pmouseX - mouseX);
}
```

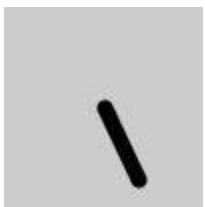
Al dibujar una línea desde la anterior posición del mouse hasta la posición actual, se revela la velocidad y la dirección del trazado. Cuando el mouse está quieto, si dibuja un punto. No obstante, al mover el mouse, se dibujan largas líneas.

```
// Dibuja un línea entre la anterior posición y la actual
//posición
void setup() {
      size(100, 100);
      strokeWeight(8);
      smooth();
}
void draw() {
      background(204);
      line(mouseX, mouseY, pmouseX, pmouseY);
}
```

Los valores de `mouseX` y `mouseY` pueden utilizarse para controla la escala, posición y rotación de los elementos del programa. Por ejemplo, pueden emplearse junto a la función `translate()`.

```
// Utilizando translate() para mover la figura
void setup() {
      size(100, 100);
      smooth();
      noStroke();
}
void draw() {
      background(126);
      translate(mouseX, mouseY);
      ellipse(0, 0, 33, 33);
}
```

Antes de utilizar los valores obtenidos por `mouseX` y `mouseY`, hay que pensar primero en la clase de parámetros que aceptan dichas funciones de transformación. Por ejemplo, la función `rotate()` solo acepta valores en radianes. Para hacer que una figura rote en 360 grados, es necesario convertir los valores de `mouseX` en valores de 0.0 a 2π. En el siguiente ejemplo, se utiliza la función `map()` para convertir dichos valores. El valor obtenido es utilizado en la función `rotate()`.



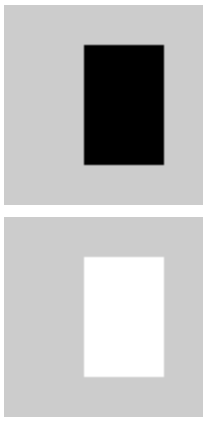
```
// Utilizando rotate() para rotar la figura
void setup() {
      size(100, 100);
      strokeWeight(8);
      smooth();
}

void draw() {
      background(204);
      float angle = map(mouseX, 0, width, 0, TWO_PI);
      translate(50, 50);
      rotate(angle);
      line(0, 0, 40, 0);
}
```

De la misma forma, se puede utilizar una estructura `IF` para reconocer individualmente diferentes sectores de la pantalla:

```
// La posición del cursor selecciona una mitad de la pantalla
void setup() {
      size(100, 100);
      noStroke();
      fill(0);
}
void draw() {
      background(204);
      if (mouseX < 50) {
            rect(0, 0, 50, 100);   //Izquierda
      } else {
            rect(50, 0, 50, 100);  //Derecha
      }
}
```

```
// La posición del cursor selecciona la izquierda, derecha o
//el centro de la pantalla
void setup() {
      size(100, 100);
      noStroke();
      fill(0);
}
void draw() {
      background(204);
      if (mouseX < 33) {
            rect(0, 0, 33, 100);          //Izquierda
      } else if ((mouseX >= 33) && (mouseX <= 66)) {
            rect(33, 0, 33, 100);         //Medio
      } else {
            rect(66, 0, 33, 100);         //Derecha
      }
}
```

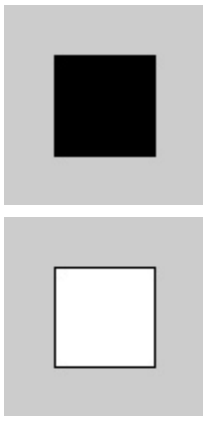
```
// La posición del cursor selecciona un cuadrante de la
//pantalla
void setup() {
      size(100, 100);
      noStroke();
      fill(0);
}
void draw() {
      background(204);
      if ((mouseX <= 50) && (mouseY <= 50)) {
            rect(0, 0, 50, 50);     //Arriba-Izquierda
      } else if ((mouseX <= 50) && (mouseY > 50)) {
            rect(0, 50, 50, 50);    //Abajo-Izquierda
      } else if ((mouseX > 50) && (mouseY < 50)) {
            rect(50, 0, 50, 50);    //Arriba-Derecha
      } else {
            rect(50, 50, 50, 50);   //Abajo-Derecha
      }
}
```



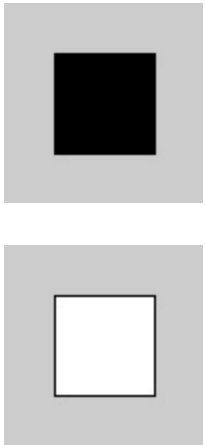
```
// La posición del cursor selecciona un área rectangular
void setup() {
      size(100, 100);
      noStroke();
      fill(0);
}
void draw() {
      background(204);
      if ((mouseX > 40) && (mouseX < 80) && (mouseY > 20)
      && (mouseY < 80)) {
            fill(255);
      } else {
            fill(0);
      }
      rect(40, 20, 40, 60);
}
```

-Botones del Mouse

Los dispositivos de entrada de las computadora, por lo general, poseen entre 2 y 3 botones, y Processing puede detectarlos. La detección de la posición del mouse, sumado a los botones, permiten utilizar al mouse como un importante dispositivo de entrada en un programa interactivo. La variable `mousePressed` devuelve un valor `true` cuando un botón del mouse es oprimido, por el contrario devuelve un `false`. Además, permite detectar que botón fue oprimido a través de la variable `mouseButton`, pudiendo ser su valor `LEFT` (izquierdo), `RIGHT` (derecho) y `CENTER` (centro), dependiendo el botón que se desee detectar. Estas variables pueden utilizarse independientes o en combinación:



```
// El cuadrado cambia a blanco cuando el botón es presionado
void setup() {
      size(100, 100);
}
void draw() {
      background(204);
            if (mousePressed == true) {
      fill(255);              //Blanco
} else {
      fill(0);                //Negro
}
rect(25, 25, 50, 50);
}
```



```
// El cuadro se vuelve negro cuando se oprime el botón
//izquierdo y blanco cuando se oprime el derecho, y gris
//cuando no se oprime ninguno.
void setup() {
      size(100, 100);
}
void draw() {
      if (mousePressed == true) {
            if (mouseButton == LEFT) {
                  fill(0);                //Negro
            } else if (mouseButton == RIGHT) {
                  fill(255);              //Blanco
            }
      } else {
            fill(126);                    //Gris
```

```
            }
            rect(25, 25, 50, 50);
      }
```

Nota: Al utilizar software que detecte dispositivos de entrada, se suele correr el riesgo de que el usuario no posea el dispositivo indicado (especialmente si se planea subir el proyecto a la web). Por ejemplo, existen usuarios que poseen un dispositivo de mouse con dos botones y otros que tienen tres botones. Es importante tener esto en mente a la hora de programar con dispositivos de entrada.

-Icono del Cursor

El cursor puede ser ocultado con la función `noCursor()`, y del mismo modo puede reemplazarse por la función `cursor()`. Cuando la función `noCursor()` se está ejecutando, el cursor se encuentra totalmente oculta, sin embargo, la posición puede conseguirse con `mouseX` y `mouseY`.

```
// Dibuja una elipse para mostrar la posición del cursor oculto
void setup() {
      size(100, 100);
      strokeWeight(7);
      smooth();
      noCursor();
}
void draw() {
      background(204);
      ellipse(mouseX, mouseY, 10, 10);
}
```

Si la función `noCursor()` se está ejecutando, el cursor estará completamente oculto hasta que se llame a la función `cursor()`:

```
// Esconde el cursor hasta que se oprima el botón del mouse
void setup() {
      size(100, 100);
      noCursor();
}
void draw() {
      background(204);
      if (mousePressed == true) {
            cursor();
      }
}
```

Además, la función `cursor()` acepta determinados parámetros que permiten cambiar el cursor determinado por defecto por otro diferente. Los parámetros auto-descriptivos son: `ARROW` (flecha), `CROSS` (cruz), `HAND` (mano), `MOVE` (mover), `TEXT` (texto) y `WAIT` (espera).

```
// Dibujar el cursor como una mano cuando el botón es oprimido
void setup() {
      size(100, 100);
      smooth();
}
void draw() {
      background(204);
      if (mousePressed == true) {
            cursor(HAND);
      } else {
            cursor(MOVE);
      }
```

```
      line(mouseX, 0, mouseX, height);
      line(0, mouseY, height, mouseY);
}
```

Las imágenes que se muestren como cursor son las que están instaladas por defecto en el ordenador, y varían entre sistemas operativos.