Unidad 24

Valores de Entrada: Eventos

Elementos que se introducen en esta Unidad:
`mousePressed(), mouseReleased(), mouseMoved(), mouseDragged()`
`keyPressed(), keyReleased()`
`loop(), redraw()`

Las funciones pueden llamar eventos que alteran el normal flujo del programa cuando una acción como un botón o una tecla es oprimida. Por lo tanto, un evento de una interrupción del flujo normal del programa. Una tecla oprimida, o la posición del mouse, es almacenada hasta que el bloque `draw()` terminé de ejecutarse. En esos casos, se puede producir un disturbio en el modo de dibujar. El código dentro de un evento se ejecuta una vez, cuando el evento ocurre. Por ejemplo, si se utilizara un evento que detecta el presionar un botón del mouse, las acciones ocurrirán solo una vez hasta que el botón sea liberado y presionado nuevamente. Esto permite que los datos que manejen los eventos del mouse o el teclado sean leídos independientemente de lo que esté ocurriendo en el bloque principal.

-Eventos del Mouse

Los eventos del mouse son `mousePressed()`, `mouseReleased()`, `mouseMoved()`, y `mouseDragged()`:

`mousePressed()`	El código dentro se ejecuta cuando un botón del mouse es oprimido.
`mouseReleased()`	El código dentro se ejecuta cuando un botón del mouse es liberado.
`mouseMoved()`	El código dentro se ejecuta cuando el cursor se mueve.
`mouseDragged()`	El código dentro se ejecuta cuando el cursor se mueve mientras un botón del mouse es oprimido

La función `mousePressed()` se ejecuta de forma diferente a la variable `mousePressed`. El valor de la variable `mousePressed` es un `true` hasta que el botón sea liberado. En cambio, el código dentro de un bloque `mousePressed()` se ejecuta cuando el botón del mouse es oprimido. En el siguiente ejemplo se puede ver como el fondo del programa cambia gradualmente su nivel de gris a medida que un botón del mouse es oprimido.

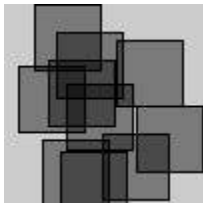
```
float gris = 0;
void setup() {
      size(100, 100);
}
void draw() {
      background(gris);
}
void mousePressed() {
      gris += 20;
}
```

En el siguiente ejemplo se ejecuta lo mismo que el ejemplo anterior, con la diferencia que lo que controla el nivel de gris es el evento de liberar el botón del mouse. La diferencia puede apreciarse manteniendo apretado el botón del mouse por un largo tiempo y así advertir que el fondo se altera cuando dicho botón es liberado.

```
float gray = 0;
void setup() {
      size(100, 100);
}
void draw() {
      background(gray);
}
```

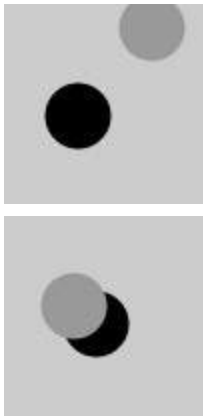
```
void mouseReleased() {
      gray += 20;
}
```

Antes de dibujar dentro de estas funciones, es importante pensar en el flujo del programa. Por ejemplo, hay círculos que son dibujados dentro de `mousePressed()`, y luego dentro del bloque `draw()` ya que retiramos la función `background()`. Sin embargo, si estuviese `background()` declarado, los elementos solo aparecerían un simple cuadro en pantalla y luego dejarían de verse.



```
void setup() {
      size(100, 100);
      fill(0, 102);
}
void draw() { }         // Bloque draw() vacío, mantiene al
void mousePressed() {   //programa ejecutándose
      rect(mouseX, mouseY, 33, 33);
}
```

El código de las funciones `mouseMoved()` y `mouseDragged()` se ejecuta cuando hay algún cambio de posición en el cursor. El código dentro de `mouseMoved()` se ejecuta al final de cada cuadro cuando la posición del mouse cambia y no es presionado ningún botón. En cambio, el código dentro de `mouseDragged()` se ejecuta cuando la posición del cursor es alterada a la vez que es oprimido el botón del mouse.



```
int dragX, dragY, moveX, moveY;
void setup() {
      size(100, 100);
      smooth();
      noStroke();
}
void draw() {
      background(204);
      fill(0);
      ellipse(dragX, dragY, 33, 33);      //Círculo Negro
      fill(153);
      ellipse(moveX, moveY, 33, 33);      //Círculo Gris
}
void mouseMoved() {    //Mueve el círculo gris
      moveX = mouseX;
      moveY = mouseY;
}
void mouseDragged() {  //Mueve el círculo negro
      dragX = mouseX;
      dragY = mouseY;
}
```

-Eventos del Teclado

Cada tecla presionada es registrada y posee dos clases de eventos, `keyPressed()` y `keyReleased()`:

`keyPressed()`	El código dentro de este bloque se ejecuta cuando un tecla es presionada
`keyReleased()`	El código dentro de este bloque se ejecuta cuando un tecla es liberada

Cada vez que una tecla es presionada podemos registrar el evento y relacionarlo a una serie de acciones. De esta forma, podemos usar `keyPressed()` para controlar valores numéricos y así utilizarlos para, por ejemplo, cambiar la posición de un rectángulo.

```
void setup() {
      size(100, 100);
      noStroke();
      fill(255, 51);
}
void draw() { }         //Bloque draw() vacío, mantiene al
void keyPressed() {     //programa ejecutándose
      int y = key - 32;
      rect(0, y, 100, 4);
}
```

Cada vez que una tecla es liberada el código dentro de `keyReleased()` es ejecutado. En el siguiente ejemplo, se dibuja una T cuando la tecla T se oprime, y desaparece cuando la tecla se suelta.


```
boolean dibujarT = false;
void setup() {
      size(100, 100);
      noStroke();
}
void draw() {
background(204);
      if (dibujarT == true) {
            rect(20, 20, 60, 20);
            rect(39, 40, 22, 45);
      }
}
void keyPressed() {
      if ((key == 'T') || (key == 't')) {
            dibujarT = true;
      }
}
void keyReleased() {
      dibujarT = false;
}
```

Los siguientes dos ejemplos trabajan con `keyPressed()` para leer y analizar los valores de entrada por el teclado combinándolos con datos del tipo `String`.


```
// Un ejemplo extremadamente simple de un editor de texto
// permite agregar y remover elementos de una línea
PFont fuente;
String letras = "";
void setup() {
      size(100, 100);
      fuente = loadFont("Consolas-24.vlw");
      textFont(fuente);
      stroke(255);
      fill(0);
}
void draw() {
      background(204);
      float cursorPosition = textWidth(letras);
      line(cursorPosition, 0, cursorPosition, 100);
      text(letras, 0, 50);
}
void keyPressed() {
      if (key == BACKSPACE) {       //Barra de Espacio
            if (letras.length() > 0) {
            letras= letras.substring(0, letras .length()-1);
```

```
            }
        } else if (textWidth(letras+key) < width){
            letras = letras+key;
        }
}
```


```
// Compara los valores de entrada del teclado
// sea “negro” o “gris” y cambia el fondo de acuerdo al valor.
// Presionar Enter o Return para activar los valores de
//entrada.
PFont fuente;
String letras = "";
int back = 102;
void setup() {
      size(100, 100);
      fuente = loadFont("Consolas-24.vlw");
      textFont(fuente);
      textAlign(CENTER);
}
void draw() {
      background(back);
      text(letras, 50, 50);
}
void keyPressed() {
      if ((key == ENTER) || (key == RETURN)) {
            letras = letras.toLowerCase();
            println(letras);       //Imprime en la consola
                                   //el valor de entrada
            if (letras.equals("negro")) {
                  back = 0;
            } else if (letras.equals("gris")) {
                  back = 204;
            }
            letras = "";           // Limpia la variable
      } else if ((key > 31) && (key != CODED)) {
      //Si la tecla es alfanumérica, la agrega al String
            letras = letras + key;
      }
}
```

-Controlando el Flujo

Los programas utilizan el bloque `draw()` para dibujar cuadro a cuadro las acciones que se pretenden tan rápido como sea posible. Con la función `frameRate()`, es posible limitar la cantidad de cuadros que ejecuta una acción cada segundo, y la función `noLoop()` es utilizada para hacer que el bloque `draw()` deje de ejecutarse constantemente. Las funciones adicionales, `loop()` y `redraw()`, proveen de mas opciones cuando se utilizan eventos del mouse y el teclado.

De este modo, se podrá ejecutar un programa que se encuentre con `noLoop()` y utilizar la función `loop()` solo cuando se requiera. Esto sirve para ahorrar una gran cantidad de recursos (especialmente si se piensa utilizar el proyecto en la web). El siguiente ejemplo ejecuta un bloque `draw()` por dos segundos cada vez que el botón del mouse es oprimido. Pasado el tiempo, el programa se pone en pausa.

```
int frame = 0;
void setup() {
      size(100, 100);
      frameRate(30);
}
void draw() {
```

```
      if (frame > 60) { // Si ya pasaron mas de 60 cuadros
            noLoop();   // desde que el mouse fue oprimido, pausar el programa
            background(0);    // y volver el fondo negro.
      } else {                // Sino, hacer el fondo gris
            background(204);                    // y dibujar líneas en la
            line(mouseX, 0, mouseX, 100);       // posición del mouse.
            line(0, mouseY, 100, mouseY);
            frame++;
      }
}
void mousePressed() {
      loop();
      frame = 0;
}
```

La función `redraw()` ejecuta el código del bloque `draw()` una vez y después detiene su ejecución. Esto es muy útil si nuestro programa no necesita ser actualizado continuamente. A continuación, se presenta un ejemplo donde el bloque `draw()` se ejecuta una vez cuando se oprime el botón del mouse:

```
void setup() {
      size(100, 100);
      noLoop();
}
void draw() {
      background(204);
      line(mouseX, 0, mouseX, 100);
}
void mousePressed() {
      redraw();         // Ejecuta el código en el draw() una vez
}
```