5.6. Gráficos vinculados a eventos de ratón

Hasta ahora se han visto pocas funciones propias de framework de GAmuza, dado que su desarrollo se dirige más a la interactividad, a partir de ahora iremos revisándolas, empezando por las funciones que trasmiten al sistema la posición del ratón en la ventana de salida, es decir, los datos relativos a las coordenadas `X` e `Y` de su posición, si se ha movido, si algún botón está presionado, ha sido soltado o es arrastrado. Las funciones para la posición del ratón son:

```
gaMouseX(), gaMouseY()     // coordenadas x e y de la posición del ratón
```

Cuando el ratón está en el lado izquierdo de la pantalla su coordenada `X` tiene como valor 0, y se incrementa conforme se desplaza hacia la derecha. Cuando el ratón está en el lado superior de la pantalla, el valor de la coordenada `Y` es `0` y se incrementa conforme va bajando. Estos datos NO los da el sistema si las funciones `gaMouseX()`y `gaMouseY()` se sitúan en el bloque `setup()` porque cuando el programa se pone en marcha la posición del ratón es `(0,0)` y el código escrito en el bloque `setup()` solo se lee una vez. Por ejemplo, para establecer relaciones de interactividad entre posición del ratón y de las formas, las funciones se sitúan en el `update()`:

```
function update()
	posX = gaMouseX()	// actualiza cada frame la posición del ratón
	posY = gaMouseY()
end

function draw()
	gaBackground(0.0, 0.5)
	ofSetColor(255)
	ofCircle(posX, posY, 50) // el centro vinculado a la posición del ratón
end
```

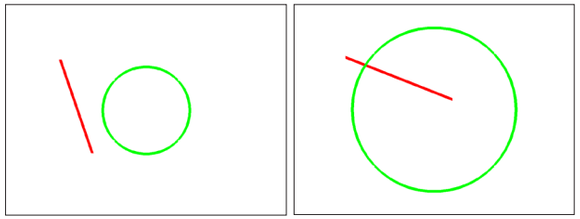
Se pueden establecer relaciones menos evidentes si se combinan operadores matemáticos que aumenten o disminuyan determinados valores, por ejemplo:

```
function draw()
   gaBackground(0.0, 0.5)
   ofSetColor(255)
   ofCircle(OUTPUT_W - gaMouseX(), gaMouseY() - OUTPUT_H, 50);
end
```

La relación inversa entre la dimensión de la ventana de salida respecto a la posición del ratón hace que el movimiento responda a direcciones con sentido contrario.

También puede aplicarse el valor de las coordenadas `X`,`Y` del ratón a las dimensiones de las formas que cambiarán de tamaño según el movimiento del ratón.

```
posX=0
posY=0

function setup()
   ofEnableSmoothing()
end
function update()
   posX= gaMouseX()
   posY= gaMouseY()
end
function draw()
   gaBackground(1.0,1.0)
   ofSetCircleResolution(50)
      ofSetColor(255,0,0)                    // color rojo
      ofSetLineWidth(10)                     // ancho línea 10px
   ofLine(posX, posY, 200, 200)              // la línea un punto fijo
                                             // y el otro en la posición del ratón
      ofSetColor(0,255,0)                    // color verde
      ofNoFill()                             // no rellenar
   ofCircle(OUTPUT_W/2, OUTPUT_H/2, posX/2)  //radio vinculado a posición X ratón
end
```



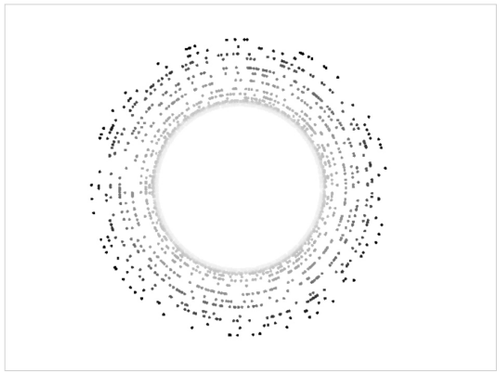
Las relaciones de tamaño entre la línea y el círculo, vinculadas a la posición del ratón, establecen una sensación como de bombeo flexible entre las formas, al estirar la línea el círculo crece.

GAmuza articula los eventos de ratón a través de bloques de función que permiten programar acciones que serán efectuadas cuando esos eventos sucedan.

```
function mouseDragged()    // Mover el ratón con el botón apretado (arrastrar)
end
function mouseMoved()      // Movimiento del ratón
end
function mousePressed()    // Botón del ratón apretado
end
function mouseReleased()   // Botón del ratón soltado
end
```

El siguiente ejemplo contiene acciones vinculadas al movimiento del ratón en el bloque `mouseMoved()`, y otras al clicar el botón, situadas en el bloque `mousePressed()`.

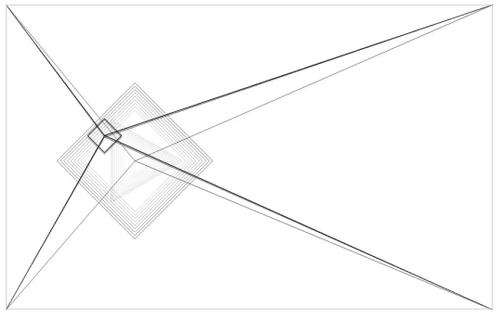

```
/*
 GAmuza 043 examples
 -------------------
 Basics/flutua_dispersa.ga
 mouseDragged y mouseMoved
*/

x = OUTPUT_W/2
y = OUTPUT_H/2
r = 2
_angle = 0.0
fluctua = 20
dispersa = 5

function setup()
   ofEnableSmoothing()
end

function draw()
   gaBackground(1.0, 0.2)
   ofSetColor(0)
   for i = 0, fluctua do
      _angle = ofRandom(TWO_PI)
      dispX = x + math.cos(_angle)*dispersa
      dispY = y + math.sin(_angle)*dispersa
      ofCircle(dispX, dispY, r*2)
   end
end

function mousePressed()
   x = gaMouseX()
   y = gaMouseY()
end

function mouseMoved()
   dispersa = gaMouseX()*0.5
   fluctua = gaMouseY()*0.5
end
```



En este caso la función `mouseMoved()` modifica el valor de las variables `fluctua` y `dispersa` . La variable `fluctua` controla el límite de repeticiones de una estructura `for` (el número de círculos pequeños que construyen el anillo) y la variable `dispersa` controla el radio del anillo de círculos generado por el `for`.

Con la función `mousePressed()` se desplaza el centro del anillo a la zona de la ventana de salida donde clique el ratón. Tras ello, si se mueve el ratón diagonalmente se verán los cambios. La transparencia del fondo permite ver el rastro de la transformación.

En el siguiente ejemplo se utiliza el movimiento `mouseMoved()`, arrastre `mouseDragged()` y soltar el botón del ratón `mouseReleased()`, para modificar el número de lados de un polígono y su posición, generando un efecto elástico, como de cuerdas de tensión, con las líneas que van de los vértices de la ventana de salida al centro del polígono.



```
/*
GAmuza 043
----------------------
mouseDragged, mouseReleased y mouseMoved
creado por Paco Fuentes
*/

x = OUTPUT_W/2
y = OUTPUT_H/2
mod = 0.1
geom = 3

function setup()
   ofEnableSmoothing()
end

function draw()
    gaBackground(1.0, 0.1)
    ofSetColor(0)
    ofNoFill()
    ofSetCircleResolution(geom)
    ofCircle(x, y, mod)
    ofSetLineWidth(ofRandomuf())
    ofLine(x, y, 0, 0)
    ofLine(OUTPUT_W, 0 , x, y)
    ofLine(0, OUTPUT_H, x, y)
    ofLine(OUTPUT_W, OUTPUT_H, x, y)
end

function mouseDragged()
    mod = mod + 10
    geom = geom + 0.1
end

function mouseReleased()
    mod = 50
end

function mouseMoved()
    x = gaMouseX()
    y = gaMouseY()
    geom = geom - 0.1
    if geom < 3 then      // si el nº de lados es menor que 3
        geom = 3          // vuelven a ser 3
    end
end
```


5.7. Gráficos vinculados a eventos de teclado

El teclado es el dispositivo habitual para escribir texto en el ordenador. Cada tecla presionada es una señal de entrada al sistema que tiene como respuesta (salida) la impresión de caracteres en el monitor. Esa señal de entrada (input) puede utilizarse para interactuar con las acciones programadas en el código.

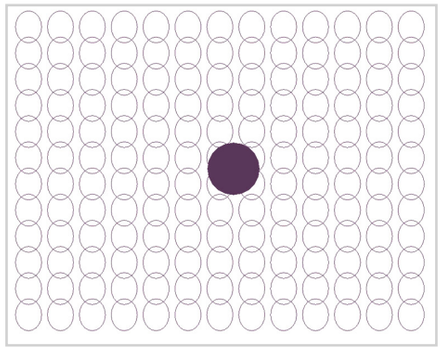
La función de GAmuza para enviar al sistema los datos de las teclas es `gaKey()` y para que el sistema sepa cuál es esa tecla, se utilizan estructuras condicionales, como:

```
if gaKey() == string.byte('g') then
        haz tal cosa
end
```

La función de Lua `string.byte` devuelve los códigos numéricos internos de los caracteres[54]. Al igual que en los eventos de ratón, GAmuza articula los eventos de teclado a través de bloques de función que permiten programar acciones que serán efectuadas cuando esos eventos sucedan.

```
function keyPressed()   //el código actúa cuando se presiona una tecla
end

function keyReleased()   //el código actúa cuando se suelta una tecla
end
```

En el siguiente ejemplo se establecen distintas composiciones en función de la tecla presionada.



```
/*
 GAmuza 043
 ----------------------
 Práctica keyboard
 activar con las teclas g, h, t
 creado por Nevena Atasanova
*/

r = 0 // ancho elipse y valor del rojo y azul
t = OUTPUT_H/2
h = 30  // radio/altura inicial círculo/elipse

function setup()
    ofEnableSmoothing ()
end

function draw()
    gaBackground(1.0,1.0)
    ofSetColor(r, 100, r)              // valores del rojo y azul serán al random
```

54 Hay que tener en cuenta que los códigos numéricos no son siempre iguales en las distintas plataformas.

```
    ofSetCircleResolution(50)
    ofCircle(OUTPUT_W/2,t,h)          // el radio será al random
    ofNoFill()
    for x=50, OUTPUT_W-50, r do       // valor incremento será al random
        for y=50, OUTPUT_H-50, h do   // valor incremento será al random
            ofEllipse(x,y,h,r)        // alto y ancho en función del n° filas y columnas
        end
    end
end

function keyReleased()
    if gaKey() == string.byte('g') then   // cada vez que se suelta la tecla g
        r = ofRandom(0,255)                // valor random para color y n° filas
    end
    if gaKey() == string.byte('h') then   // cada vez que se suelta la tecla h
        h = ofRandom(0,400)                // valor random para radio y n° columnas
    end
    if gaKey() == string.byte('t') then   // cada vez que se suelta la tecla t
        t = t-3                            // círculo central baja
    end
end
```