



## Manejo de la librería Graphics y draw

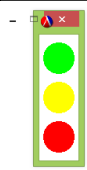
### Crear una Ventana (de n X m)

	<pre>(require (lib "graphics")) (open-graphics) (define Miventana (open-viewport "Ventana de Inicio" 600 600))</pre>
---	--

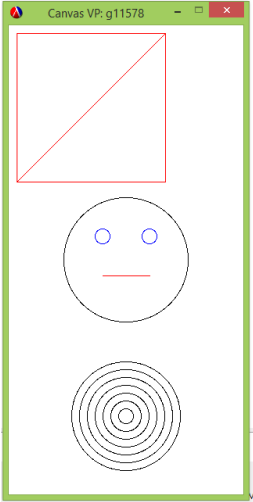
### Insertar una imagen en una ventana

	<pre>(require (lib "graphics")) (open-graphics) ; creamos una ventana (define Miventana (open-viewport "Ventana de Inicio" 600 600)) ; insertamos la imagen en la ventana en la posición que queramos ((draw-pixmap Miventana) "mario1.jpg" (make-posn 0.0 0.0))</pre>
---	--


### Ejemplos con la Librería Draw (Ejemplo 1)

	<pre>; draw.ss : funciones basicas (require (lib "draw.ss" "htdp")) ; Dibujo de un semaforo ; dimensions of traffic light semaforo (define ANCHURA 50) (define ALTURA 160) (define RADIO 20) (define DISTANCIA 10) ;; las posiciones de los bulbos (define X-BULBS (quotient ANCHURA 2)) (define Y-ROJO (+ DISTANCIA RADIO)) (define Y-AMARILLO (+ Y-ROJO DISTANCIA (* 2 RADIO))) (define Y-VERDE (+ Y-AMARILLO DISTANCIA (* 2 RADIO))) ;; dibujar la luz con el bulbo rojo girado n (start ANCHURA ALTURA) (draw-solid-disk (make-posn X-BULBS Y-ROJO) RADIO 'green) (draw-solid-disk (make-posn X-BULBS Y-AMARILLO) RADIO 'yellow) (draw-solid-disk (make-posn X-BULBS Y-VERDE) RADIO 'red)</pre>
---	---

## Ejemplos con la Librería Draw (Ejemplo 2)

	<pre> ;draw.ss : funciones basicas  (require (lib "draw.ss" "htdp")) ;(open-graphics) ; no necesita (start 300 600)  ;Triangulo (draw-solid-line (make-posn 10 10) (make-posn 200 10) 'red) (draw-solid-line (make-posn 10 10) (make-posn 10 200) 'red) (draw-solid-line (make-posn 200 10) (make-posn 10 200)'red)  ;Cuadrado (draw-solid-line (make-posn 10 10) (make-posn 200 10) 'red) (draw-solid-line (make-posn 10 10) (make-posn 10 200) 'red) (draw-solid-line (make-posn 200 10) (make-posn 200 200)'red) (draw-solid-line (make-posn 10 200) (make-posn 200 200) 'red)  ;carita (draw-circle (make-posn 150 300) 80 'black) (draw-circle (make-posn 180 270) 10 'blue) (draw-circle (make-posn 120 270) 10 'blue) (draw-solid-line (make-posn 120 320) (make-posn 180 320) 'red)  ; tiro al blanco (define(figura radio)   (if (&lt; radio 80)     (begin       (display (draw-circle (make-posn 150 500) radio 'black))       (figura (+ 10 radio))     )     (display "fin")   ) )  (figura 10) </pre>
---	---

## Manejo de Cuadrantes usando graphics

	<pre> ;(require (lib "graphics.ss" "graphics")) (require (lib "graphics")) (open-graphics) (define v (open-viewport "prueba" 800 800))  (define posx 50) (define posy 50) (define ancho 300) (define alto 300) (define color (make-rgb 0.1 0.1 0.1)) (define n 50) </pre>
---	---

```

(define dormido 0.1)

(define (pintar-cuadro posx posy ancho alto color)
  (begin
    ((draw-solid-rectangle v) (make-posn posx posy) ancho alto color)
    ((draw-rectangle v) (make-posn posx posy) ancho alto (make-rgb 0 0
0))))

(define (escoger-cuadro n posx posy ancho alto color)
  (cond
    [(= n 1)(pintar-cuadro posx posy ancho alto color)]
    [(= n 2)(pintar-cuadro (+ posx ancho) posy ancho alto color)]
    [(= n 3)(pintar-cuadro (+ posx ancho) (+ posy alto) ancho alto color)]
    [(= n 4)(pintar-cuadro posx (+ posy alto) ancho alto color)]))

(define (rotar n dormido cont cuadro posx posy ancho alto r g b)
  (if (<= cont n)
    (begin
      (sleep dormido)
      (if (= cuadro 1)
        (begin
          (escoger-cuadro 1 posx posy ancho alto (make-rgb r g b))
          (escoger-cuadro 4 posx posy ancho alto (make-rgb 1 1 1))
        )
        (begin
          (escoger-cuadro cuadro posx posy ancho alto (make-rgb r g b))
          (escoger-cuadro (- cuadro 1) posx posy ancho alto (make-rgb 1 1
1))
        ))
      (if (= cuadro 4)
        (rotar n dormido (+ cont 1) 1 posx posy ancho alto (/ (random 100)
100) (/ (random 100) 100) (/ (random 100) 100))
        (rotar n dormido (+ cont 1) (+ cuadro 1) posx posy ancho alto (/
(random 100) 100) (/ (random 100) 100) (/ (random 100) 100))))))

(define (empezar)
  (rotar n dormido 1 1 posx posy ancho alto 0.1 0.1 0.1))

(empezar)

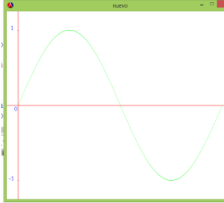
(close-graphics)

```

## Tablero de ajedrez

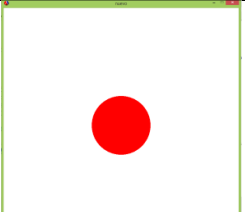
	<pre>(require (lib "graphics.ss" "graphics")) (open-graphics) (define w (open-viewport "HOLA" 680 680)) (define(lineal x y)   (if(&lt;= x 580)     (begin       ((draw-solid-rectangle w)(make-posn x y) 40 40 "black")       ((draw-solid-rectangle w)(make-posn (+ x 40)y) 40 40 "blue")       (lineal(+ x 80 )y)     )     )   ) (define(lineaP x y)   (if(&lt;= x 580)     (begin       ((draw-solid-rectangle w)(make-posn x y) 40 40 "blue")       ((draw-solid-rectangle w)(make-posn (+ x 40)y) 40 40 "black")       (lineaP(+ x 80)y)     )     )   ) (define(cuadroc y)   (if(&lt;= y 580)     (begin       (lineal 20 y)       (lineaP 20 (+ y 40))       (cuadroc (+ y 80))     )     )   ) (cuadroc 20)</pre>
---	--

## Función Seno

	<pre>;(require (lib "graphics.ss" "graphics")) (require (lib "graphics")) (open-graphics)  (define pantallax 600) (define pantallay 500) (define w (open-viewport "nuevo" pantallax pantallay))  ;pone los puntos de la grafica del seno(x), se ajusta a las dimensiones de la pantalla (px,py). desplx equivale a lo que cambia x por cada pixel en el mismo eje. pixelx es una contador que señala el pixel de la pantalla en el eje x donde se pone el punto.  (define (grafica-seno px py x desplx pixelx)   (if (&lt;= x (* 2 pi))</pre>
---	---

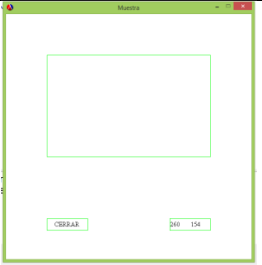
	<pre> (begin   ((draw-line w) (make-posn pixelx (- (/ py 2) (* (sin x) (* py 0.4))))   (make-posn pixelx (- (/ py 2) (* (sin x) (* py 0.4)))) (make-rgb 0 1 0))   (grafica-seno px py (+ x desplx) desplx (+ pixelx 1))))  ; pinta grafica del seno(x) dentro de las coordenadas de pantalla dadas por (px,py). (define (principal px py)   (begin     ((draw-line w) (make-posn (/ px 20) 0) (make-posn (/ px 20) py)     (make-rgb 1 0 0))     ((draw-line w) (make-posn 0 (/ py 2)) (make-posn px (/ py 2)) (make-     rgb 1 0 0))     ((draw-string w) (make-posn (- (/ px 20) 10)(+ (/ py 2) 15)) "0"     (make-rgb 0 0 1))     ((draw-string w) (make-posn (- (/ px 20) 20) (* py 0.1)) "1" (make-     rgb 0 0 1))     ((draw-line w) (make-posn (- (/ px 20) 1) (* py 0.1)) (make-posn (+ (/     px 20) 1) (* py 0.1)) (make-rgb 1 0 0))     ((draw-string w) (make-posn (- (/ px 20) 25) (* py 0.9)) "-1" (make-     rgb 0 0 1))     ((draw-line w) (make-posn (- (/ px 20) 1) (* py 0.9)) (make-posn (+ (/     px 20) 1) (* py 0.9)) (make-rgb 1 0 0))      (grafica-seno px py 0 (/ (* 2 pi) (* px 0.9)) (/ px 20))     ))  (principal pantallax pantallay) </pre>
--	--

### Manejo del Ratón (Ejemplo1)

	<pre> (require (lib "graphics")) (open-graphics) (define w (open-viewport "nuevo" 800 800))  (define (distancia p1 p2)   (begin     (newline)     (display (posn-x p1))     (newline)     (display (posn-y p1))     (newline)     (display (posn-x p2)) </pre>
---	--

	<pre> (newline) (display (posn-y p2)) (newline) (sqrt (+ (sqr (- (posn-x p2) (posn-x p1))) (sqr (- (posn-y p2) (posn-x p1)))))) ))  (define (cara2 p1 p2 radio descriptor bandera)   (begin     (cond       ((left-mouse-click? descriptor) (display "Boton izquierdo"))       ((middle-mouse-click? descriptor) (display "Boton medio"))       ((right-mouse-click? descriptor) (display "Boton derecho"))     )     (newline)     (if (&lt;= (distancia p1 p2) radio)       (begin         (display "Dentro de la cara")         (cara1 p1 radio (get-mouse-click w) 0))       (begin         (display "fuera de la cara")         ;(cara1 p1 radio (get-mouse-click w) 1)        ))     ))  (define (cara1 p1 radio descriptor bandera)   (begin     (newline)     (display bandera)     (newline)      (if (= 0 bandera)       (cara2 p1 (mouse-click-posn descriptor) radio descriptor bandera))))  (define (caricatura)   (begin     ((draw-solid-ellipse w) (make-posn 300 300) 200 200 "red")     (cara1 (make-posn 400 400) 100 (get-mouse-click w) 0))     (close-graphics))  (caricatura) </pre>
--	---

## Manejo del Ratón (Ejemplo 2)

	<pre>;(require (lib "graphics.ss" "graphics")) (require (lib "graphics")) (open-graphics) (define w (open-viewport "Muestra" 600 600))  (define (MostrarPosicion posi)   ((draw-solid-rectangle w) (make-posn 401 501) 98 28    "white")   ((draw-string w) (make-posn 401 520) (number-&gt;string     (posn-x posi)))   ((draw-string w) (make-posn 451 520) (number-&gt;string     (posn-y posi)))   )  (define (EsperarClick w)  (define c (ready-mouse-click w)) (if (not c)   (begin     (MostrarPosicion (query-mouse-posn w))     (EsperarClick w)   )   (begin     (ValidarClick c)   )   )   )  (define (MostrarNuevaVentana w)   ((draw-rectangle w) (make-posn 100 500) 100 30    "green")   ((draw-rectangle w) (make-posn 400 500) 100 30    "green")   ((draw-rectangle w) (make-posn 100 100) 400 250    "green")   ((draw-string w) (make-posn 118 520) "CERRAR")   (EsperarClick w)   (close-graphics)   )  (MostrarNuevaVentana w) (define (Lapiz w po1 po2)  (define c (ready-mouse-release w))  (define po (query-mouse-posn w)) (if (not c)   (begin     (if (and (and (&lt;= (posn-x po) (posn-x po2)) (&lt;=</pre>
---	---

```

(posn-y po) (posn-y po2))) (and (>= (posn-x po) (posn-x
po1)) (>= (posn-y po) (posn-y po1))))
(begin
  ((draw-pixel w) po "green")
  (Lapiz w po1 po2)
)
(begin
  (EsperarClick w)
)
)
)
(begin
  (EsperarClick w)
)
)
)

(define (ValidarClick c)
  (define po1 (make-posn 100 500))
  (define po2 (make-posn 200 530))
  (define po3 (make-posn 100 100))
  (define po4 (make-posn 500 350))

  (define po (mouse-click-posn c))
  (if (and (and (<= (posn-x po) (posn-x po2)) (<= (posn-
y po) (posn-y po2))) (and (>= (posn-x po) (posn-x po1))
(>= (posn-y po) (posn-y po1))))
    (begin
      (display "Good Bye!")
      (close-viewport w)
    )
    (begin
      (if (and (and (<= (posn-x po) (posn-x po4)) (<=
(posn-y po) (posn-y po4))) (and (>= (posn-x po) (posn-x
po3)) (>= (posn-y po) (posn-y po3))))
        (begin
          (Lapiz w po3 po4)
        )
        (begin
          (EsperarClick w)
        )
      )
    )
  )
)

```