

Por la pestaña “Lenguaje” → “Añadir paquete de enseñanza” .

Se escoge del directorio, “C:\Archivos de programa\PLT\teachpack\htdp” el paquete de enseñanza: “C:\Archivos de programa\PLT\teachpack\htdp\image.ss.”

Images image.ss

Este paquete de enseñanza provee primitivas para la construcción y manipulación de imágenes. Estas funciones crean formas básicas. El modo puede ser con el string “solid” ó el símbolo ‘solid, indicando que estas formas se rellenaran ó el símbolo ‘outline o el string “outline” que significa que la forma es hueca. El color de las imágenes puede ser símbolos (vgr: ‘red), string’s (Vgr:”red”) o estructuras RGB (**R**ed, **G**reen, **B**lue), como (make-color 0 0 255))

La mayoría de las imágenes tienen su origen en la mitad, a excepción de las líneas y textos que tienen su inicio en su esquina inferior izquierda.

FUNCIONES

image	rectangle	int ₁ int ₂ modo color
		Parámetros: int ₁ : Ancho en pixels en x int ₂ : Alto en pixeles en y modo: “solid” ó ‘solid “outline” ó ‘outline color: ‘red “red” (make-color 255 0 0))

Ejemplos:


(rectangle 100 10 'outline 'red) (rectangle 200 10 "outline" "red") (rectangle 40 40 'outline 'black) (rectangle 40 40 "solid" 'green)	
---	--

image	circle	int modo color
		Parámetros: int: Valor del radio modo: “solid” ó ‘solid “outline” ó ‘outline color: ‘red “red” (make-color 255 0 0))

Ejemplos:



(circle 30 "outline" 'red)	
(circle 30 "solid" 'red)	

image	ellipse	int ₁ int ₂ modo color
		Parámetros: int ₁ : Ancho int ₂ : Alto modo: "solid" ó 'solid "outline" ó 'outline color: 'red "red" (make-color 255 0 0))

Ejemplos:



(ellipse 20 20 "outline" 'red)	
(ellipse 40 10 "solid" (make-color 255 0 0))	
	

image	triangle	int modo color
		Parámetros: int: Lado del triangulo equilátero modo: "solid" ó 'solid "outline" ó 'outline color: 'red "red" (make-color 255 0 0))

Ejemplos:

(triangle 20 "outline" 'red)	
(triangle 40 "solid" (make-color 0 255 0))	
	

image	line	int ₁ int ₂ color
		Construye una línea del punto (0,0) al (int ₁ , int ₂) Parámetros: int ₁ : Ancho int ₂ : Alto color: 'red "red" (make-color 255 0 0))

Ejemplos:


(line 10 20 'red)	
(line 0 30 (make-color 0 255 0))	
(line 30 0 (make-color 0 0 255))	

image	add-line	image int ₁ int ₂ int ₃ int ₄ color
		<p>Adiciona una línea a una imagen existente, dibujándola entre dos puntos.</p> <p>Parámetros:</p> <p>int₁ int₂: punto x,y de la imagen seleccionada (origen)</p> <p>int₃ int₄: punto x,y a donde llegara la nueva linea adicionada</p> <p>color: 'red</p> <p>“red”</p> <p>(make-color 255 0 0))</p>

Ejemplos:

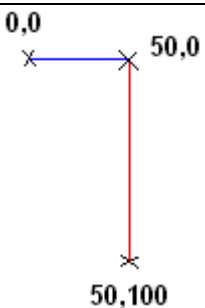
<p>; definiremos a <i>im</i> como una <i>línea</i></p> <p>(define im (line 50 0 'blue))</p> <p>(add-line im 50 0 50 100 'red)</p>	
---	--

image	text	string int color
		<p>Crea una imagen del texto (string), con el tamaño y color especificados.</p> <p>Parámetros:</p> <p>string: Texto a dibujar</p> <p>int₁: Tamaño</p> <p>color: 'red</p> <p>“red”</p> <p>(make-color 255 0 0))</p>

Ejemplos:

(text "Hola amigos como están" 20 'red)	Hola amigos como están
(define cadena (string #\U #\T #\P))	
(text cadena 20 'red)	

image	overlay	image₁ image₂ image₃..... Adiciona los pixeles de la image ₃ sobre la image ₂ y sobre la image ₁ .
--------------	----------------	--

Ejemplos:






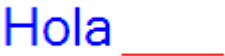
(define trazo (line 50 0 'red)) (define cadena (string #\H #\o #\l #\a)) (define equi (triangle 30 'solid 'blue)) (define texto (text cadena 20 'red)) (overlay equi texto trazo)	 Note que el texto queda encima del triángulo
(overlay texto trazo equi)	 Note que el texto queda debajo del triángulo

image	Overlay/xy	image₁ int₁ int₂ image₂ Adiciona los pixeles de la image ₂ sobre la image ₁ alejándolo (int ₁ , int ₂) del inicio (0,0) para texto y líneas y/o centro para las otras imágenes.
--------------	-------------------	--

Ejemplos:

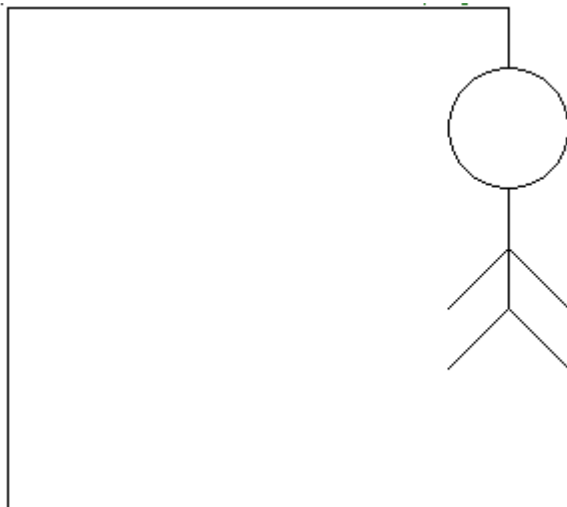
(define trazo (line 50 0 'red)) (define cadena (string #\H #\o #\l #\a)) (define texto (text cadena 20 'blue)) (overlay/xy texto 0 0 trazo)	
(overlay/xy texto 0 15 trazo)	
(overlay/xy texto 0 30 trazo)	
(overlay/xy texto 60 30 trazo)	

Ejercicio: Dibujar para el juego del ahorcado, la horca y el muñequito completo, con las funciones de image.ss

Solución: Construiremos paso a paso la figura

```
(define horca-base (line 50 0 'black))
(define horca-paral (line 0 250 'black))
(define horca-horizontal (line 250 0 'black))
(define vertical (line 0 30 'black))
(define horca (overlay/xy horca-paral 0 0 horca-horizontal))
(define horca (overlay/xy horca 250 0 vertical))
(define horca (overlay/xy horca 0 250 horca-base))
(display " ")
(define cabeza (circle 30 "outline" 'black))
(define horca-cabeza (overlay/xy horca 250 60 cabeza))
(define horca-cuello (overlay/xy horca-cabeza 250 90 vertical))
(define diagonal-1 (line 30 30 'black))
(define horca-mano1 (overlay/xy horca-cuello 250 120 diagonal-1))
(define horca-mano2 (add-line horca-mano1 250 120 220 150 'black))
(define horca-cuerpo (overlay/xy horca-mano2 250 120 vertical))
(define horca-pierna1 (add-line horca-cuerpo 250 150 220 180 'black))
(define horca-colgado (overlay/xy horca-pierna1 250 150 diagonal-1))
horca-colgado
```

Al ejecutar este programa tendremos:



Ejercicio: Construir un programa en DrScheme, tal que lea un string y dibuje tantas rayitas como caracteres tenga el string leído.

1. Análisis:

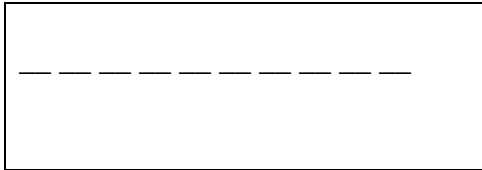
- Leer la cadena y guardarlo en una variable
- Contar el número de caracteres del string leído utilizando la función (string-length string)

c. Construir un ciclo de 1 hasta (string-length string) y dibujar las rayitas, adicionando a la rayita inicial nuevas rayitas, separadas en proporción al ciclo en el que vaya y recordando dejar un(1) espacio en blanco.

2. Contrato: Se pide hacer un programa que lea un string, y se dibujen tantas rayitas como caracteres tenga el string leído, utilizando las funciones interconstruidas(built-in) del paquete de enseñanza “image.ss”

3. Ejemplos:

a. Si el string es “Hola mundo”, deberá aparecer una pantalla, así:



b. Si el string es “UTP”, deberá aparecer una pantalla, así:



4. Plantillas

Tipo de dato a devolver	Nombre de la función	Parámetros
image	rayitas	cadena : De tipo string, es el string del cual vamos a dibujar tantas rayitas como caracteres tenga. Esta función invoca a otra auxiliar.
image	raya	función : La función “line” que construyo una línea desde el punto (0,0) al punto (30,0)
image	ray	imagen : La imagen de la rayita a duplicar control-ciclo : inicia en uno(1) fin-ciclo : va hasta la longitud del string leído

5. Desarrollo

```
(define (rayitas cadena)
  (ray raya 1 (string-length cadena) )
)

(define raya (line 30 0 'black) )

(define (ray imagen control-ciclo fin-ciclo)
  (if (<= control-ciclo fin-ciclo)
      (ray (overlay/xy imagen (* 60 (- control-ciclo 1)) 0 raya) (+ control-ciclo 1) fin-ciclo)
      imagen
  )
)

(rayitas (read))
```