

PLT Miscellaneous Libraries: Reference Manual

PLT (scheme@plt-scheme.org)

360

Released November 2006

Copyright notice

Copyright ©1996-2006 PLT

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Library General Public License, Version 2 published by the Free Software Foundation. A copy of the license is included in the appendix entitled "License."

Send us your Web links

If you use any parts or all of the PLT Scheme package (software, lecture notes) for one of your courses, for your research, or for your work, we would like to know about it. Furthermore, if you use it and publicize the fact on some Web page, we would like to link to that page. Please drop us a line at *scheme@plt-scheme.org*. Evidence of interest helps the DrScheme Project to maintain the necessary intellectual and financial support. We appreciate your help.

Contents

1	Miscellaneous Libraries	1
2	Viewport Graphics	2
2.1	Basic Commands	2
2.2	Position Operations	2
2.3	Color Operations	3
2.4	Draw, Clear and Flip Operations	3
2.4.1	Viewports	4
2.4.2	Pixels	4
2.4.3	Lines	4
2.4.4	Rectangles	4
2.4.5	Ellipses	5
2.4.6	Polygons	5
2.4.7	Strings	6
2.4.8	Pixmap	6
2.5	World Operations	7
2.6	Miscellaneous Operations	7
2.7	An Example	7
2.8	A More Complicated Example	8
2.9	Protecting Graphics Operations	8
2.10	Mouse Operations	8
2.11	Keyboard Operations	9
2.12	Flushing	10
2.13	Unitized Graphics	10

3 Turtles	11
3.1 Turtles	11
3.2 Value Turtles	13
License	15
Index	19

1. Miscellaneous Libraries

This manual documents miscellaneous libraries distributed with DrScheme.

2. Viewport Graphics

The viewport graphics library is a relatively simple toolbox of graphics commands. The library is not very powerful; it is intended as a simplified alternative to MrEd's full graphical toolbox.

The graphics library originated as SIXlib, a library of X Windows commands available within Chez Scheme at Rice University. The functionality of that library has been reproduced (with backward compatibility) in this version.

To use the viewport graphics library in a PLT language, load it via `require`:

```
(require (lib "graphics.ss" "graphics"))
```

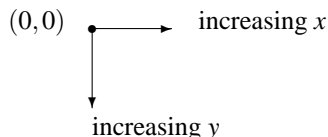
which loads the **graphics.ss** library from the **graphics** collection. All of the names defined in this chapter will then be available.

2.1 Basic Commands

- `(open-graphics)`
Initializes the library's graphics routines. It must be called before any other graphics operations.
- `(close-graphics)`
Closes all of the windows and until `open-graphics` is called again, no graphics routines will work.
- `(open-viewport name horiz vert)`
Takes a string *name* and integers *horiz* and *vert* and creates a new window called *name*. The window is *horiz* pixels wide and *vert* pixels high. For backward compatibility, a single `posn` value (see below) can be submitted in the place of *horiz* and *vert*. `open-viewport` returns a viewport descriptor.
- `(open-pixmap name horiz vert)`
Like `open-viewport`, but the resulting viewport is not displayed on the screen. Offscreen pixmaps are useful for executing a sequence of drawing commands and displaying them all at once with `copy-viewport`.
Offscreen pixmaps are also useful in conjunction with `viewport-ζsnip` (see below). This allows functions to compute with graphical objects and view the graphics when results are returned to the interactions window.
- `(close-viewport viewport)`
Takes a viewport descriptor. It removes the viewport from the screen and makes subsequent operations dealing with the viewport illegal.

2.2 Position Operations

A position is a pixel location within a viewport. The upper-left corner is pixel $(0,0)$ and the orientation of the position coordinates within a viewport is as follows:



- `(make-posn x y)`
Takes two integers and returns a position with the specified x and y coordinates.
- `(posn-x p)`, `(posn-y p)`
Return the x and y coordinates, respectively, of a position.
- `(posn? v)`
Reports whether v is a position.
- `((get-pixel viewport) p)`
Returns the color of the pixel at position p in $viewport$; 0 denotes white and 1 denotes not white.
- `((get-color-pixel viewport) p)`
Returns an RGB value for color of the pixel at position p in $viewport$.
- `((test-pixel viewport) color)`
Returns the color that will actually be used if $color$ is used to draw.

2.3 Color Operations

A color can be represented in three ways: as a color index (an integer in 0 to 299, inclusive), as a color name string, or as a `rgb` value. All drawing functions which take a color argument accept colors in any form. An `rgb` value is assigned to an index with `change-color`.

- `(make-rgb red green blue)`
Takes three values in the range 0 (dark) to 1 (bright) and returns an `rgb` (a color).
- `(rgb-red color)`
`(rgb-blue color)`
`(rgb-green color)`
Return the red, green, and blue components, respectively, of a color.
- `(rgb? v)`
Reports whether v is a color.
- `(change-color index rgb)`
Changes the color at $index$ in the color table to the color specified in rgb . Only the first twenty-one indices are initialized; a color index should not be used until it has been initialized.
- `(default-display-is-color?)`
Returns `#t` if the default display screen for viewports is in color or `#f` otherwise.

2.4 Draw, Clear and Flip Operations

These are the basic graphics operations for drawing to a viewport. Each function takes a viewport as its argument and returns a function operating within that viewport. Further arguments, if any, are curried. For example, `(draw-line viewport)` returns a function, that can then be applied to the proper arguments to draw a line in the viewport corresponding to viewport descriptor $viewport$. An example follows.

Where “draw-” commands make pixels black, “clear-” commands make them white.

Where “draw-” commands make pixels black, a “flip-” commands cause them to change.

2.4.1 Viewports

- `((draw-viewport viewport) color)`
Takes a viewport descriptor. It returns a function that colors the entire contents of *viewport*. The optional *color* argument defaults to black.
- `((clear-viewport viewport))`
Takes a viewport descriptor. It returns a function that whitens the entire contents of *viewport*.
- `((flip-viewport viewport))`
Takes a viewport descriptor. It returns a function that flips the contents of *viewport*.
- `(copy-viewport source-viewport destination-viewport)`
Takes two viewport descriptors. It copies the *source-viewport* into the *destination-viewport*.

2.4.2 Pixels

- `((draw-pixel viewport) p color)`
Takes a viewport descriptor. It returns a function that draws a pixel in *viewport* at the specified position. The optional *color* argument defaults to black.
- `((clear-pixel viewport) p)`
Takes a viewport descriptor. It returns a function that clears a pixel in *viewport* at the specified position.
- `((flip-pixel viewport) p)`
Takes a viewport descriptor. It returns a function that flips a pixel in *viewport* at the specified position.

2.4.3 Lines

- `((draw-line viewport) p1 p2 color)`
Takes a viewport descriptor. It returns a function that draws a line in the *viewport* connecting positions *p1* and *p2*. The optional *color* argument defaults to black.
- `((clear-line viewport) p1 p2)`
Takes a viewport descriptor. It returns a function that clears a line in *viewport* connecting positions *p1* and *p2*.
- `((flip-line viewport) p1 p2)`
Takes a viewport descriptor. It returns a function that flips a line in *viewport* connecting positions *p1* and *p2*.

2.4.4 Rectangles

- `((draw-rectangle viewport) posn width height color)`
Takes a viewport descriptor. It returns a function that draws a rectangle border in the *viewport* with the top-left of the rectangle at the position *posn* and with sides *width* across and *height* tall. The optional *color* argument defaults to black.
- `((clear-rectangle viewport) posn width height)`
Takes a viewport descriptor. It returns a function that clears a rectangle border in the *viewport* with the top-left of the rectangle at the position *posn* and with sides *width* across and *height* tall. The optional *color* argument defaults to black.

- `((flip-rectangle viewport) posn width height color)`
Takes a viewport descriptor. It returns a function that flips a rectangle border in the *viewport* with the top-left of the rectangle at the position *posn* and with sides *width* across and *height* tall. The optional *color* argument defaults to black.
- `((draw-solid-rectangle viewport) posn width height color)`
Takes a viewport descriptor. It returns a function that paints a solid rectangle in the *viewport* with the top-left of the rectangle at the position *posn* and with sides *width* across and *height* tall. The optional *color* argument defaults to black.
- `((clear-solid-rectangle viewport) posn width height)`
Takes a viewport descriptor. It returns a function that erases a solid rectangle in the *viewport* with the top-left of the rectangle at the position *posn* and with sides *width* across and *height* tall. The optional *color* argument defaults to black.
- `((flip-solid-rectangle viewport) posn width height color)`
Takes a viewport descriptor. It returns a function that flips a solid rectangle in the *viewport* with the top-left of the rectangle at the position *posn* and with sides *width* across and *height* tall. The optional *color* argument defaults to black.

2.4.5 Ellipses

- `((draw-ellipse viewport) posn width height color)`
Takes a viewport descriptor. It returns a function that draws an ellipse border in the *viewport*. The *posn*, *width*, and *height* arguments are as in `draw-rectangle`; the ellipse is inscribed within the specified rectangle. The optional *color* argument defaults to black.
- `((clear-ellipse viewport) posn width height)`
Takes a viewport descriptor. It returns a function that clears an ellipse border in the *viewport*. The *posn*, *width*, and *height* arguments are as in `clear-rectangle`; the ellipse is inscribed within the specified rectangle. The optional *color* argument defaults to black.
- `((flip-ellipse viewport) posn width height color)`
Takes a viewport descriptor. It returns a function that flips an ellipse border in the *viewport*. The *posn*, *width*, and *height* arguments are as in `flip-rectangle`; the ellipse is inscribed within the specified rectangle. The optional *color* argument defaults to black.
- `((draw-solid-ellipse viewport) posn width height color)`
Takes a viewport descriptor. It returns a function that paints a solid ellipse in the *viewport*. The *posn*, *width*, and *height* arguments are as in `draw-rectangle`; the ellipse is inscribed within the specified rectangle. The optional *color* argument defaults to black.
- `((clear-solid-ellipse viewport) posn width height)`
Takes a viewport descriptor. It returns a function that erases a solid ellipse in the *viewport*. The *posn*, *width*, and *height* arguments are as in `clear-rectangle`; the ellipse is inscribed within the specified rectangle. The optional *color* argument defaults to black.
- `((flip-solid-ellipse viewport) posn width height color)`
Takes a viewport descriptor. It returns a function that flips a solid ellipse in the *viewport*. The *posn*, *width*, and *height* arguments are as in `flip-rectangle`; the ellipse is inscribed within the specified rectangle. The optional *color* argument defaults to black.

2.4.6 Polygons

- `((draw-polygon viewport) posn-list posn color)`
Takes a viewport descriptor. It returns a function that draws a polygon border in the *viewport* using

posn-list for the polygon vertices and *posn* as an offset for the polygon. The optional *color* argument defaults to black.

- `((clear-polygon viewport) posn-list posn)`
Takes a viewport descriptor. It returns a function that erases a polygon border in the *viewport* using *posn-list* for the polygon vertices and *posn* as an offset for the polygon.
- `((flip-polygon viewport) posn-list posn)`
Takes a viewport descriptor. It returns a function that flips a polygon border in the *viewport* using *posn-list* for the polygon vertices and *posn* as an offset for the polygon.
- `((draw-solid-polygon viewport) posn-list posn color)`
Takes a viewport descriptor. It returns a function that paints a solid polygon in the *viewport* using *posn-list* for the polygon vertices and *posn* as an offset for the polygon. The optional *color* argument defaults to black.
- `((clear-solid-polygon viewport) posn-list posn)`
Takes a viewport descriptor. It returns a function that erases a solid polygon in the *viewport* using *posn-list* for the polygon vertices and *posn* as an offset for the polygon.
- `((flip-solid-polygon viewport) posn-list posn)`
Takes a viewport descriptor. It returns a function that flips a solid polygon in the *viewport* using *posn-list* for the polygon vertices and *posn* as an offset for the polygon.

2.4.7 Strings

- `((draw-string viewport) p string color)`
Takes a viewport descriptor. It returns a function that draws a string at a specified location in the *viewport*. The lower left of the string begins at *p*. The optional *color* argument defaults to black.
- `((clear-string viewport) p string)`
Takes a viewport descriptor. It returns a function that clears a string at a specified location in *viewport*. The lower left of the string begins at *p*.
- `((flip-string viewport) p string)`
Takes a viewport descriptor. It returns a function that flips a string at a specified location in *viewport*. The lower left of the string begins at *p*.

2.4.8 Pixmaps

- `((draw-pixmap-posn filename type) viewport) posn color)`

Draws a pixmap into *viewport* with its upper left corner at position *posn*. The *type* is an optional symbol, one of 'gif, 'gif/mask, 'xbm, 'xpm, 'bmp, 'pict, 'unknown, or 'unknown/mask, and defaults to 'unknown/mask. If *type* is 'unknown or 'unknown/mask, then the content of the file is examined to determine the type. All formats are supported on all platforms, except 'pict which is only supported under Mac OS X. The 'gif/mask and 'unknown/mask types draw the bitmap with a transparent background if *filename* refers to a GIF file with a transparent background.

The argument *color* is only used when the pixmap is black and white. In that case, the color is used instead of black in the drawn image.

- `((draw-pixmap viewport) filename p color)`
Draws a pixmap into *viewport w* with its upper left corner at position *p*. If *color* is not #f it is passed to `set-viewport-pen` with the viewport. It defaults to #f.

- `((save-pixmap viewport) filename type)`
Saves the current content of *viewport* to *filename*. The *type* is an optional symbol, one of 'x_{bm}', 'x_{pm}', 'b_{mp}' (Windows only), or 'p_{ict}' (Mac OS X only); the default is 'x_{pm}'.

2.5 World Operations

Every canvas comes with an associated world. A client program can set the world, start the world's clock, stop the world's clock, and deal with tick events (the clock ticks) and keyboard inputs (keyevents).

- `((init-world viewport) X)`
Takes a viewport descriptor. It returns a function whose input becomes the initial value of the world associated with this canvas.
- `((set-on-tick-event viewport) number unary procedure)`
Takes a viewport descriptor. It returns a function whose first input is a number and the second one is a function from worlds to worlds. The number determines how frequently the clock ticks. The given function is called for every clock tick on the current world; the result becomes the next world.
- `((stop-tick viewport))`
Takes a viewport descriptor. It returns a function of no arguments that can stop the clock for this canvas's world.
- `((set-on-key-event viewport) unary procedure)`
Takes a viewport descriptor. It returns a function whose input becomes the keyevent callback function. This callback consumes the latest keyevent and the current world and a keyevent; it produces the next world.

2.6 Miscellaneous Operations

- `((get-string-size viewport) string)`
Takes a viewport descriptor. It returns a function that returns the size of a string as a list of two numbers: the width and height.
- `(viewport->snip viewport)`
Takes a viewport descriptor. It returns an object that can be inserted into an editor buffer to display the current image in the viewport. (Subsequent drawing to the viewport does not affect the snip's image.)
When snips are the results of computations in the interactions window, DrScheme will print show the contents of the viewport, right in the interactions window.
- `(viewport-dc viewport)`
Takes a viewport descriptor. It returns an object that can be used with the primitive MrEd toolbox functions to draw into the viewport's on-screen representation (if any). Mirror all such drawing to the result of `(viewport-offscreen-dc viewport)`, too.
- `(viewport-offscreen-dc viewport)`
Takes a viewport descriptor. It returns an object that can be used with the primitive MrEd toolbox functions to draw into the viewport's off-screen representation. Mirror all such drawing to the result of `(viewport-dc viewport)`, too.

2.7 An Example

```
(open-graphics)
; nothing appears to happen, but the library is initialized...

(define w (open-viewport "practice" 300 300))
```

```

;; viewport appears

((draw-line w) (make-posn 30 30) (make-posn 100 100))
;; line appears

(close-viewport w)
;; viewport disappears

(close-graphics)
;; again, nothing appears to happen, but
;; unclosed viewports (if any) would disappear

```

2.8 A More Complicated Example

The use of multiple viewports, viewport descriptors, drawing operations for multiple viewports is as easy as the use of a single viewport:

```

(open-graphics)
(let* ( ;; w1 and w2 are viewport descriptors for different windows
      [w1 (open-viewport "viewport 1" 300 300)]
      [w2 (open-viewport "viewport 2" 200 500)]
      ;; d1 and d2 are functions that draw lines in different viewports
      [d1 (draw-line w1)]
      [d2 (draw-line w2)])
  ;; draws a line in viewport labeled "viewport 1"
  (d1 (make-posn 100 5) (make-posn 5 100))
  ;; draws a line in viewport labeled "viewport 2"
  (d2 (make-posn 100 100) (make-posn 101 400)))

;; we no longer have access to viewports 1 and 2,
;; since their descriptors did not escape the let
(close-graphics)
;; removes the viewports

```

2.9 Protecting Graphics Operations

To guarantee the proper closing of viewports in cases of errors, especially when a program manages several viewports simultaneously, a programmer should use `dynamic-wind`:

```

(let ([w (open-viewport "hello" 100 100)])
  (dynamic-wind
   ;; what we want to happen first: nothing
   void
   ;; the main program (errors constrained to this piece)
   (lambda () (draw-pixel 13)) ; an error
   ;; what we would like to happen, whether the main program finishes
   ;; normally or not
   (lambda () (close-viewport w))))

```

2.10 Mouse Operations

The graphics library contains functions that determine where the mouse is, if there are any clicks, etc. The functions `get-mouse-click` and `ready-mouse-click` first return a “mouse-click descriptor,” and then other functions

take the descriptor and return the mouse's position, which button was pushed, etc. Mouse clicks are buffered and returned in the same order in which they occurred. Thus, the descriptors returned by `get-mouse-click` and `ready-mouse-click` may be from clicks that occurred long before these functions were called.

- `(get-mouse-click viewport)`
Takes a viewport descriptor and returns a mouse click descriptor. It returns the next mouse click in the *viewport*, waiting for a click if necessary.
- `(ready-mouse-click viewport)`
Takes a viewport descriptor and returns either a mouse click descriptor, or else `#f` if none is available. Unlike the previous function, *ready-mouse-click* returns immediately.
- `(ready-mouse-release viewport)`
Takes a viewport descriptor and returns either a click descriptor from a mouse-release (button-up) event, or else `#f` if none is available.
- `(query-mouse-posn viewport)`
Takes a viewport descriptor and returns either the position of the mouse cursor within the *viewport*, or else `#f` if the cursor is currently outside the *viewport*.
- `(mouse-click-posn mouse-click)`
Takes a mouse click descriptor and returns the position of the pixel where the click occurred.
- `(left-mouse-click? mouse-click)`
Takes a mouse click descriptor and returns `#t` if the click occurred with the left mouse button, or else `#f`.
- `(middle-mouse-click? mouse-click)`
Similar to `left-mouse-click?`.
- `(right-mouse-click? mouse-click)`
Similar to `left-mouse-click?`.

2.11 Keyboard Operations

The graphics library contains functions that report key presses from the keyboard. The functions `get-key-press` and `ready-key-press` return a “key-press descriptor,” and then `key-value` takes the descriptor and returns a character or symbol (usually a character) representing the key that was pressed. Key presses are buffered and returned in the same order in which they occurred. Thus, the descriptors returned by `get-key-press` and `ready-key-press` may be from presses that occurred long before these functions were called.

- `(get-key-press viewport)`
Takes a viewport descriptor and returns a key press descriptor. It returns the next key press in the *viewport*, waiting for a click if necessary.
- `(ready-key-press viewport)`
Takes a viewport descriptor and returns either a key press descriptor, or else `#f` if none is available. Unlike the previous function, *ready-key-press* returns immediately.
- `(key-value key-press)`
Takes a key press descriptor and returns a character or special symbol for the key that was pressed. For example, the Enter key generates `#\return`, and the up-arrow key generates `'up`. For a complete list of possible return values, see *PLT MrEd: Graphical Toolbox Manual*.

2.12 Flushing

- `(viewport-flush-input viewport)`

As noted above, key presses and mouse clicks are buffered. `viewport-flush-input` takes a viewport descriptor and empties the input buffer of mouse and keyboard events.

2.13 Unitized Graphics

To use a unitized version of the graphics library (see *PLT MzLib: Libraries Manual* for more information on units), get the signatures `graphics^`, `graphics:posn-less^`, and `graphics:posn^` with:

```
(require (lib "graphics-sig.ss" "graphics"))
```

The `graphics^` signature includes all of the names defined in this chapter. The `graphics:posn-less^` signature contains everything except the `posn` structure information, and `graphics:posn^` contains only the `posn` structure.

To obtain `graphics@`, which imports `mred^` (all of the `MrEd` classes, functions, and constants) and exports `graphics^`:

```
(require (lib "graphics-unit.ss" "graphics"))
```

The **`graphics-posn-less-unit.ss`** library provides `graphics-posn-less@`, which imports `graphics:posn^` in addition to `MrEd`.

3. Turtles

3.1 Turtles

There are two ways to use the turtles in DrScheme. You can use it as a TeachPack (see the DrScheme manual for details of TeachPacks) or as a library. Use the **turtles.ss** TeachPack.

In the MrEd language or in a module, load turtles with

```
(require (lib "turtles.ss" "graphics"))
```

The following are the turtle functions:

- `(turtles b)` shows and hides the turtles window based on the boolean `b`. The parameter `b` is optional; if it is left out, it toggles the state of the turtles.
- `(move n)` moves the turtle `n` pixels.
- `(draw n)` moves the turtle `n` pixels and draws a line on that path.
- `(erase n)` moves the turtle `n` pixels and erases along that path.
- `(move-offset h v)`, `(draw-offset h v)`, `(erase-offset h v)` are just like `move`, `draw` and `erase`, except they take a horizontal and vertical offset from the turtle's current position.
- `(turn theta)` turns the turtle `theta` degrees counter-clockwise.
- `(turn/radians theta)` turns the turtle `theta` radians counter-clockwise.
- `(clear)` erases the turtles window.

Turtles also defines these syntactic forms:

- `(split E)` spawns a new turtle where the turtle is currently located. In order to distinguish the two turtles, only the new one evaluates the expression `E`. For example, if you start with a fresh turtle-window and type:

```
(split (turn/radians (/ pi 2)))
```

you will have two turtles, pointing at right angles to each other. To see that, try this:

```
(draw 100)
```

You will see two lines. Now, if you evaluate those two expression again, you will have four turtles, etc

- `(split* E ...)` is similar to `(split E ...)`, except it creates as many turtles as there are expressions and each turtles does one of the expression. For example, to create two turtles, one pointing at $\pi/2$ and one at $\pi/3$, evaluate this:

```
(split* (turn/radians (/ pi 3)) (turn/radians (/ pi 2)))
```

- `(tprompt E...)` provides a way to limit the splitting of the turtles. Before the expression `E` is run, the state of the turtles (how many, their positions and headings) is "checkpointed," then `E` is evaluated and the state of the turtles is restored, but all drawing that may have occurred during execution of `E` remains.

For example, if you do this:

```
(tprompt (draw 100))
```

the turtle will move forward 100 pixels, draw a line there and then be immediately put back in its original position. Also, if you do this:

```
(tprompt (split (turn/radians (/ pi 2))))
```

the turtle will split into two turtles, one will turn 90 degrees and then the turtles will be put back into their original state – as if the split never took place.

The fern functions below demonstrate more advanced use of `tprompt`.

In the file **turtle-examples.ss** in the **graphics** library of your PLT distribution, you will find these functions and values defined, as example turtle programs. (The file is located in the **graphics** subdirectory of the **collects** subdirectory of the PLT distribution).

- `(regular-poly sides radius)` draws a regular poly centered at the turtle with sides *sides* and with radius *radius*.
- `(regular-polys sides s)` draws *s* regular polys spaced evenly outwards with sides *sides*.
- `(radial-turtles n)` places 2^n turtles spaced evenly pointing radially outward
- `(spaced-turtles n)` places 2^n turtles pointing in the same direction as the original turtle evenly spaced in a line.
- `(spokes)` draws some spokes, using `radial-turtles` and `spaced-turtles`
- `(spyro-gyra)` draws a spyro-gyra reminiscent shape
- `(neato)` as the name says...
- `(graphics-bexam)` draws a fractal that came up on an exam I took.
- `serp-size` a constant which is a good size for the `serp` procedures
- `(serp serp-size)`, `(serp-nosplit serp-size)` draws the Sierpinski triangle in two different ways, the first using `split` heavily. After running the first one, try executing `(draw 10)`.
- `koch-size` a constant which is a good size for the `koch` procedures
- `(koch-split koch-size)`, `(koch-draw koch-size)` draws the same koch snowflake in two different ways.
- `(lorenz a b c)` watch the lorenz "butterfly" attractor with initial values *a* *b* and *c*.
- `(lorenz1)` a good setting for the lorenz attractor
- `(peano1 peano-size)`

This will draw the Peano space-filling curve, using `split`.

- `(peano2 peano-size)`

This will draw the Peano space-filling curve, without using `split`.

- `fern-size` a good size for the fern functions

- `(fern1 fern-size)` You will probably want to point the turtle up before running this one, with something like:

```
(turn/radians (- (/ pi 2)))
```

- `(fern2 fern-size)` a fern – you may need to backup a little for this one.

3.2 Value Turtles

There are two ways to use the turtles in DrScheme. You can use it as a TeachPack (see the DrScheme manual for details of TeachPacks) or as a library. Use the **value-turtles.ss** TeachPack.

In the MrEd language or in a module, load turtles with

```
(require (lib "value-turtles.ss" "graphics"))
```

The value turtles are a variation on the turtles library. Rather than having just a single window where each operation changes the state of that window, in this library, the entire turtles window is treated as a value. This means that each of the primitive operations accepts, in addition to the usual arguments, a turtles window value and instead of returning nothing, returns a turtles window value.

The following are the value turtle functions:

- `(turtles number number [number number number])` creates a new turtles window. The first two arguments are the width and height of the turtles window. The remaining arguments specify the x,y position of the initial turtle and the angle. The default to a turtle in the middle of the window, pointing to the right.
- `(move n turtles)` moves the turtle `n` pixels, returning a new turtles window.
- `(draw n turtles)` moves the turtle `n` pixels and draws a line on that path, returning a new turtles window.
- `(erase n turtles)` moves the turtle `n` pixels and erases along that path, returning a new turtles window.
- `(move-offset h v turtles)`, `(draw-offset h v turtles)`, `(erase-offset h v turtles)` are just like `move`, `draw` and `erase`, except they take a horizontal and vertical offset from the turtle's current position.
- `(turn theta turtles)` turns the turtle `theta` degrees counter-clockwise, returning a new turtles window.
- `(turn/radians theta)` turns the turtle `theta` radians counter-clockwise, returning a new turtles window.
- `(merge turtles turtles)`

The `split` and `tprompt` functionality provided by the imperative turtles implementation aren't needed for this, since the turtles window is itself a value.

Instead, the `merge` accepts two turtles windows and combines the state of the two turtles windows into a single window. The new window contains all of the turtles of the previous two windows, but only the line drawings of the first turtles argument.

In the file **value-turtles-examples.ss** in the **graphics** library of your PLT distribution, you will find these functions and values defined, as example turtle programs. (The file is located in the **graphics** subdirectory of the **collects** subdirectory of the PLT distribution).

It contains a sampling of the examples from the normal turtles implementation, but translated to use `merge` and the `values turtles`.

License

GNU Library General Public License

Version 2, June 1991

Copyright (C) 1991 Free Software Foundation, Inc.

675 Mass Ave, Cambridge, MA 02139, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the library GPL. It is numbered 2 because it goes with version 2 of the ordinary GPL.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries

themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a “work based on the library” and a “work that uses the library”. The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

GNU LIBRARY GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called “this License”). Each licensee is addressed as “you”.

A “library” means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The “Library”, below, refers to any such software library or work which has been distributed under these terms. A “work based on the Library” means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term “modification”).

“Source code” for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library’s complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

- a) The modified work must itself be a software library.
- b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
- c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
- d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

- a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
- b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
- c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
- d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
- b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.
10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients’ exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.
14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Index

`#/return`, 9

Butterfly Attractor, 12

`change-color`, 3

`clear`, 11

`clear-ellipse`, 5

`clear-line`, 4

`clear-pixel`, 4

`clear-polygon`, 6

`clear-rectangle`, 4

`clear-solid-ellipse`, 5

`clear-solid-polygon`, 6

`clear-solid-rectangle`, 5

`clear-string`, 6

`clear-viewport`, 4

`close-graphics`, 2

`close-viewport`, 2

`copy-viewport`, 4

`default-display-is-color?`, 3

`draw`, 11, 13

`draw-ellipse`, 5

`draw-line`, 4

`draw-offset`, 11, 13

`draw-pixel`, 4

`draw-pixmap`, 6

`draw-pixmap-posn`, 6

`draw-polygon`, 5

`draw-rectangle`, 4

`draw-solid-ellipse`, 5

`draw-solid-polygon`, 6

`draw-solid-rectangle`, 5

`draw-string`, 6

`draw-viewport`, 4

`erase`, 11, 13

`erase-offset`, 11, 13

Fern Fractal, 13

`flip-ellipse`, 5

`flip-line`, 4

`flip-pixel`, 4

`flip-polygon`, 6

`flip-rectangle`, 5

`flip-solid-ellipse`, 5

`flip-solid-polygon`, 6

`flip-solid-rectangle`, 5

`flip-string`, 6

`flip-viewport`, 4

`get-color-pixel`, 3

`get-key-press`, 9

`get-mouse-click`, 9

`get-pixel`, 3

`get-string-size`, 7

`graphics`

- `simple`, 2

graphics.ss, 2

`graphics:posn-less^`, 10

`graphics:posn^`, 10

`graphics@`, 10

`graphics^`, 10

`init-world`, 7

`key-value`, 9

Koch Snowflake, 12

`left-mouse-click?`, 9

Lorenz Attractor, 12

`make-posn`, 3

`make-rgb`, 3

`merge`, 13

`middle-mouse-click?`, 9

`mouse-click-posn`, 9

`move`, 11, 13

`move-offset`, 11, 13

`open-graphics`, 2

`open-pixmap`, 2

`open-viewport`, 2

Peano space-filling curve, 12

`posn-x`, 3

`posn?`, 3

`query-mouse-posn`, 9

`ready-key-press`, 9

`ready-mouse-click`, 9

`ready-mouse-release`, 9

`rgb-red`, 3

`rgb?`, 3

`right-mouse-click?`, 9

`save-pixmap`, 7

Serpinski Triangle, 12

`set-on-key-event`, 7

`set-on-tick-event`, 7

split, 11
split*, 11
stop-tick, 7

test-pixel, 3
tprompt, 12
turn, 11, 13
turn/radians, 11, 13
Turtles
 Value, 13
turtles, 11, 13
turtles.ss, 11

Value Turtles, 13
value-turtles.ss, 13
viewport, 2
viewport->snip, 7
viewport-dc, 7
viewport-flush-input, 10
viewport-offscreen-dc, 7