Tickit::Window

Mouse events

press	A mouse button has been pressed down on this cell
drag_start	The mouse was moved while a button was held, and was initially in the given cell
drag	The mouse was moved while a button was held, and is now in the given cell
drag_outside	The mouse was moved outside of the window that handled the "drag_start" event, and is still being dragged.
drag_drop	A mouse button was released after having been moved, while in the given cell
drag_stop	The drag operation has finished
release	A mouse button was released after being pressed

Attributes

parent	Returns the parent window; i.e. the window on which "make_sub" or "make_float" was called to create this one
subwindows	Returns a list of the subwindows of this one. They are returned in order, highest first.
root	Returns the root window
term	Returns the Tickit::Term instance of the terminal on which this window lives. Returns the Tickit instance with which this window is associated.
is_visible	Returns true if the window is currently visible.
is_focused	Returns true if this window currently has the input focus
rect	Returns a Tickit::Rect containing representing the window's extent relative to its parent
top bottom left right	Position relative to parent
abs_top abs_left	Top left co-ordinate in the window (from root window)
cols lines	Window size
pen	Returns the current Tickit::Pen object associated with this window

Methods

make_sub $top, $left, $lines, $cols	Constructs a new sub-window of the given geometry, and places it at the end of the child window list; below any other siblings.
make_hidden_sub $top, $left, $lines, $cols	Constructs a new sub-window like make_sub, but the window starts initially hidden. This avoids having to call hide separately afterwards.
make_float $top, $left, $lines, $cols	Constructs a new sub-window of the given geometry, and places it at the start of the child window list; above any other siblings.
make_popup $top, $left, $lines, $cols	Constructs a new floating popup window starting at the given coordinates relative to this window, with root window as parent
raise_to_front raise lower lower_to_back	Moves the order of the window in its parent relative to its siblings.
close	Removes the window from its parent, clearing event handlers, recursively closing any child windows
show	Makes the window visible, will invoke the on_expose handler
hide	Makes the window invisible
resize $lines, $cols	Change the size of the window.
reposition $top, $left	Move the window relative to its parent.
change_geometry $top, $left, $lines, $cols	A combination of "resize" and "reposition", to atomically change all the coordinates of the window.
set_on_geom_changed $on_geom_changed	Set the callback to invoke whenever the window is resized or repositioned, will be passed the window object
set_on_key $on_key	Set the callback to invoke whenever a key is pressed while this window or children have the input focus, will be passed type, str and mod
set_on_mouse $on_mouse	Set the callback to invoke whenever a mouse event is received within the window's rectangle.
set_on_expose $on_expose	Set the callback to invoke whenever a region of the window is exposed, will be passed the window and Tickit::Rect representing the exposed region.
expose $rect	Marks the given region of the window as having been exposed
set_on_focus $on_refocus	Set the callback to invoke whenever the window gains or loses focus, will be passed the window and true/false
set_expose_after_scroll $expose_after_scroll	If set to a true value, the scrollrect method will expose the region of the window that requires redrawing
set_pen $pen	Replace the current Tickit::Pen object for this window with a new one - undef means new blank pen
getpenattr $attr	Returns a single attribue from the current pen
get_effective_pen	Returns a new Tickit::Pen containing the effective pen attributes for the window, combined by those of all its parents.
get_effective_penattr $attr	Returns the effective value of a pen attribute. This will be the value of this window's attribute if set, or the effective value of the attribute from its parent.
goto $line, $col	Moves the cursor to the given 0-based position within the window.
print $text, $pen | %attrs	Print the given text to the terminal at the current cursor position, returning a Tickit::StringPos object giving the total count of string printed
erasech $count, $moveend, $pen | %attrs	Erase $count columns forwards. If $moveend is true, the cursor will be placed at the end of the erased region. Returns a Tickit::StringPos object.
clearrect $rect, $pen | %attrs	Erase the content of the window within the given Tickit::Rect.
scrollrect $top, $left, $lines, $cols, $downward, $rightward, $pen | %attrs	Attempt to scroll the rectangle of the window defined by the first four parameters by an amount given by the latter two.
scroll $downward, $rightward	A shortcut for calling "scrollrect" on the entire region of the window.
cursor_at $line, $col	Sets the position in the window at which the terminal cursor will be placed when this window has focus, see also take_focus.
cursor_shape $shape	Sets the shape that the terminal cursor will have if this window has focus. See TERMCTL_CURSORSHAPE_* constants in Tickit::Term.
take_focus	Causes this window to take the input focus, and updates the cursor position to the stored active position given by cursor_at.
focus $line, $col	A convenient shortcut combining "cursor_at" with "take_focus"; setting the focus cursor position and taking the input focus.
restore	Restore the state of the terminal to its idle state. Places the cursor back at the focus position, and restores the pen.
clearline $line	Erase the entire content of one line of the window
clear	Erase the entire content of the window and reset it to the current background colour.

Tickit::Widget

Attributes

style_classes	Returns a list of the style class names this Widget has.
get_style_text	A shortcut to calling "get_style_values" for a single key called "text".
pen	Returns the widget's Tickit::Pen. Modifying an attribute of the returned object results in the widget being redrawn if the widget has a window associated.
window	Returns the current window of the widget, if one has been set using "set_window".
parent	Returns the current container widget

Methods

focus_next_before focus_next_after	Requests the focus move to the next or previous focusable widget in display order.
set_style_tag $tag, $value	Sets the (boolean) state of the named style tag, see also "style_reshape_keys", "style_reshape_textwidth_keys" and "style_redraw_keys"
get_style_values @keys	Returns a list of values for the given keys of the currently-applied style. For more detail see the Tickit::Style documentation. Returns just one value in scalar context.
get_style_pen $prefix	A shortcut to calling "get_style_values" to collect up the pen attributes, and form a Tickit::Pen::Immutable object from them, using ${prefix}_* if a prefix is given
set_style %defs	Apply the given hash to the widget style.
set_window $window	Sets the Tickit::Window for the widget to draw on, or undef to remove the current window
set_parent $parent	Sets the parent widget; pass "undef" to remove the parent. $parent, if defined, must be a subclass of Tickit::ContainerWidget.
resized	Provided for subclasses to call when their size requirements have or may have changed. Informs the parent that the widget may require a differently-sized window.
redraw	Mark this widget as needing a redraw
set_pen $pen	Set a new "Tickit::Pen" object. This is stored by reference; changes to the pen will be reflected in the rendered look of the widget. The same pen may be shared by more than one widget; updates will affect them all.
take_focus	Calls take_focus on the Widget's underlying Window, if present, or stores that the window should take focus when one is eventually set by set_window

Subclass Methods

render_to_rb	Called to redraw the widget's content to the given Tickit::RenderBuffer. Will be passed the clipping rectangle region to be rendered; the method does not have to render any content outside of this region.
reshape	Optional. Called after the window geometry is changed. Useful to distribute window change sizes to contained child widgets.
lines	Called to enquire on the requested window for this widget. It is possible that the actual allocated window may be larger, or smaller than this amount.
window_gained	Optional. Called by "set_window" when a window has been set for this widget.
window_lost	Optional. Called by "set_window" when "undef" has been set as the window for this widget. The old window object is passed in.
on_key	Optional. If provided, this method will be set as the "on_key" callback for any window set on the widget. By providing this method a subclass can implement widgets that respond to user input. It receives the same arguments as the underlying window "on_key" event.
on_mouse	Optional. If provided, this method will be set as the "on_mouse" callback for any window set on the widget. By providing this method a subclass can implement widgets that respond to user input. If receives the same arguments as the underlying window "on_mouse" event.
on_style_changed_values	Optional. Will be called by set_style_tag when style keys may have changed values. Will be passed as name => [old,new]. See style_reshape_keys, style_redraw_keys.
CAN_FOCUS	Optional, normally false. If this constant method returns a true value, the widget is allowed to take focus using the "take_focus" method. It will also take focus automatically if it receives a mouse button 1 press event.
KEYPRESSES_FROM_STYLE	Optional, normally false. If this constant method returns a true value, then <key_$key> will be invoked if found, instead of on_key ("<Space>" can be used for the space key)

Tickit

Constructor

Takes the following named arguments:

term_in	IO handle for terminal input. Will default to "STDIN"
term_out	IO handle for terminal output. Will default to "STDOUT"
root	If defined, sets the root widget using "set_root_widget" to the one specified.

Attributes

term	Returns the underlying Tickit::Term object
cols	Number of columns on the terminal
lines	Query the current size of the terminal. Will be cached and updated on receipt of "SIGWINCH" signals.
rootwin	Returns the root Tickit::Window.

Methods

later $code	Runs the given CODE reference at some time soon in the future. It will not be invoked yet, but will be invoked at some point before the next round of input events are processed.

timer
$mode, $amount, $code
	Runs the given CODE reference at some fixed point in time in the future. $mode must be either the string "at", or "after"; and specifies that $amount gives either the absolute epoch time, or the delay relative to now, respectively. Fractions are supported to a resolution of microseconds.
	`timer(at => $epoch, $code )`
	`timer(after => $delay, $code )`

bind_key $key, $code	Installs a callback to invoke if the given key is pressed, overwriting any previous callback for the same key, pass $code=undef to remove: `$code->( $tickit, $key )`
set_root_widget $widget	Sets the root widget for the application's display. This must be a subclass of Tickit::Widget.
setup_term	Set up the screen and generally prepare to start running
teardown_term	Shut down the screen after running
tick	Run a single round of IO events. Does not call "setup_term" or "teardown_term".
run	Calls the "setup_term" method, then processes IO events until stopped, by the "stop" method, "SIGINT", "SIGTERM" or the "Ctrl-C" key. Then runs the "teardown_term" method, and returns.
stop	Causes a currently-running "run" method to stop processing events and return.

Tickit::RenderBuffer

Constructor

Takes the following named arguments:

lines	number of lines in the buffer area
cols	number of cols in the buffer area

Line types

LINE_SINGLE	A single, thin line
LINE_DOUBLE	A pair of double, thin lines
LINE_THICK	A single, thick line

Cap types

CAP_START	The start of the line fills the entire cell
CAP_END	Line end fills the entire cell
CAP_BOTH	Start and end both fill their respective cells

Methods

cols	Returns the size of the buffer area
col	Returns the current position of the virtual cursor, or "undef" if it is not set.
save	Pushes a new state-saving context to the stack, which can later be returned to by the "restore" method.
savepen	Like save but stores the pen only, restore with restore.
restore	Pops and restores a saved state previously created with "save".
clip $rect	Restricts future drawing operations to the given clipping rectangle (effect is cumulative).
translate $downward, $rightward	Applies a translation to all future drawing operations
reset	Removes any pending changes, undefines the virtual cursor position, resets the clipping rectangle, and clears the stack of saved state.
clear $pen	Resets every cell in the buffer to an erased state. A shortcut to calling "erase_at" for every line.
goto $line, $col	Sets the position of the virtual cursor.
setpen $pen	Sets the rendering pen to use for drawing operations
skip_at $line, $col, $len	Sets the range of cells given to a skipped state. No content will be drawn here, nor will any content existing on the window be erased. Initially, or after calling "reset", all cells are set to this state.
skip $len	Sets the range of cells at the virtual cursor position to a skipped state, and updates the position.
skip_to $col	Sets the range of cells from the virtual cursor position until before the given column to a skipped state, and updates the position to the column. If the position is already past this column then the cursor is moved backwards and no buffer changes are made.
text_at $line, $col, $text, $pen	Sets the range of cells starting at the given position, to render the given text in the given pen.
text $text, $pen	Sets the range of cells at the virtual cursor position to render the given text in the given pen, and updates the position.
erase_at $line, $col, $len, $pen	Sets the range of cells given to erase with the given pen.
erase $len, $pen	Sets the range of cells at the virtual cursor position to erase with the given pen, and updates the position.
erase_to $col, $pen	Erase all cells to the target column, leaving the cursor at that column
eraserect $rect, $pen	Sets the range of cells given by the rectangle to erase with the given pen.
hline_at $line, $startcol, $endcol, $style, $pen, $caps	Draws a horizontal line between the given columns (both are inclusive), in the given line style, with the given pen.
vline_at $startline, $endline, $col, $style, $pen, $caps	Draws a vertical line between the centres of the given lines (both are inclusive), in the given line style, with the given pen.
char_at $line, $col, $codepoint, $pen	Sets the given cell to render the given Unicode character (as given by codepoint number, not character string) in the given pen
flush_to_window $win	Renders the stored content to the given Tickit::Window. After this, the buffer will be cleared and reset back to initial state.

Tickit::Style

Style files

```
WidgetClass.styleclass:tag {
 key1: "value 1";
 key2: 123;
 key3: true;
 <Enter>: activate;
}
```

Subclassing

If a Widget class is subclassed and the subclass does not declare "use Tickit::Style" again, the subclass will be transparent from the point of view of style. Any style applied to the base class will apply equally to the subclass, and the name of the subclass does not take part in style decisions.
If the subclass does "use Tickit::Style" again then the new subclass will be a distinct widget type for style purposes, and it will require its own new set of base style definitions.

style_definition $tags, %definition	In addition to any loaded stylesheets, the widget class itself can provide style information, via the "style_definition" function. It provides a definition equivalent to a stylesheet definition with no style class, optionally with a single set of tags. To supply no tags, use the special string "base"
style_reshape_keys @keys	Declares that the given list of keys are somehow responsible for determining the shape of the widget. If their values are changed, the "reshape" method is called.
style_reshape_textwidth_keys @keys	Declares that the given list of keys contain text, the "textwidth()" of which is used to determine the shape of the widget. If their values are changed such that the "textwidth()" differs, the "reshape" method is called.
style_redraw_keys @keys	Declares that the given list of keys are somehow responsible for determining the look of the widget, but in a way that does not determine the size. If their values are changed, the "redraw" method is called. Between them these three methods may help avoid "Tickit::Widget" classes from needing to override the "on_style_changed_values" method.

Additional functions

These must be fully-qualified with Tickit::Style::

load_style $string	Loads definitions from a stylesheet given in a string. Definitions will be merged with existing definitions in memory, with new values overwriting existing values.
load_style_file $path	Loads definitions from a stylesheet file given by the path. Definitions will be merged the same way as "load_style".
on_style_load $code	Adds a CODE reference to be invoked after either "load_style" or "load_style_file" are called. This may be useful to flush any caches or invalidate any state that depends on style information.