





## Frames

Operation	Keystroke	Function	Note
<b>Emacs Frames</b> See <a href="#">Quick intro</a> <a href="#">Emacs Wiki Glossary</a> <a href="#">Emacs Lisp Frames</a>		<ul style="list-style-type: none"> <li>Emacs calls frames what modern graphical Operating Systems call “windows”, or more specifically “OS windows”.</li> <li>Emacs supports multiple frames, both when Emacs works in graphical mode and in text terminal mode.               <ul style="list-style-type: none"> <li>In terminal mode, only one frame is visible at any given time, the other frames are hidden. The current frame number is shown on the mode line.                   <ul style="list-style-type: none"> <li>Using multiple frames is a powerful way to keep track of development contexts, with each frame having its own set of windows.</li> </ul> </li> </ul> </li></ul>	
<b>Open this PDF file.</b> See also: <a href="#">↶ Help/Info</a>	<b>&lt;f11&gt; F &lt;f1&gt;</b>	<b>(pel-help-pdf</b> &optional OPEN-WEB-PAGE)	Open the <a href="#">↶ Frames</a> local PDF. If the prefix argument (like <b>C-u</b> or <b>M--</b> ) is used, then it opens the remote GitHub hosted raw PDF instead. If the <b>pel-flip-help-pdf-arg</b> user-option is set it's the other way around.
<a href="#">↶ Customize</a> PEL frame control	<b>&lt;f11&gt; F &lt;f2&gt;</b>	<b>(pel-customize-pel</b> &optional OTHER-WINDOW)	Customize PEL frame management support. <ul style="list-style-type: none"> <li>If OTHER-WINDOW is non-nil (use <b>C-u</b>) , display in other window.</li> </ul>
<a href="#">↶ Customize</a> Emacs frame control	<b>&lt;f11&gt; F &lt;f3&gt;</b>	<b>(pel-customize-library</b> &optional OTHER-WINDOW)	Customize Emacs frame management support.
Enter/Exit full screen	<b>&lt;f11&gt; &lt;f11&gt;</b>	<b>(pel-toggle-frame-fullscreen)</b>	Toggle frame fullscreen mode on/off in graphics mode.
	<ul style="list-style-type: none"> <li>In Terminal mode, issue error to show how to use the OS keystrokes to toggle the fullscreen mode. For example it will tell you to use <b>⌘-C-f</b> when Emacs is running inside a Terminal.app frame.  Unfortunately the information is only provided for macOS Terminal.app and iTerm.app.</li> </ul>		
	✂ Standard GNU Emacs normally binds <b>&lt;f11&gt;</b> to (toggle-frame-fullscreen). But PEL use <b>&lt;f11&gt;</b> as a prefix key, therefore it rebinds it.		
<b>Set Frame Font</b>	With Emacs running in graphics mode, you can change the font of all windows with the menu-set-font command.		
<b>Change font of current Frame</b> See also: <a href="#">↶ Faces/Fonts</a>	<b>&lt;f11&gt; F F</b>	<b>(menu-set-font)</b>	Interactively select a font and make it the default on all frames. <ul style="list-style-type: none"> <li>Set the default on both the existing and future frames in the current session. It is not persistent, so next time you start Emacs the default font is used.</li> </ul>
<b>Move to another graphical frame using cursors</b>	The following commands move point to another graphical Emacs frame existing in the direction selected by the cursor key.  <b>framemove</b>  PEL installs and activates this when the <b>pel-use-framemove</b> user option is set to <b>t</b> . Also available in the <a href="#">Emacs Wiki framemove</a> .  These four commands commands only work in <b>graphics mode</b> .		
<b>Move to graphical frame above</b>	<ul style="list-style-type: none"> <li><b>&lt;f11&gt; S-&lt;up&gt;</b></li> <li><b>&lt;Esc&gt; S-&lt;up&gt;</b></li> </ul>	<b>(fm-up-frame)</b>	Move point to frame above current frame (if one exists and is located there).
<b>Move to graphical frame below</b>	<ul style="list-style-type: none"> <li><b>&lt;f11&gt; S-&lt;down&gt;</b></li> <li><b>&lt;Esc&gt; S-&lt;down&gt;</b></li> </ul>	<b>(fm-down-frame)</b>	Move point to frame below current frame (if one exists and is located there).
<b>Move to graphical frame at right</b>	<ul style="list-style-type: none"> <li><b>&lt;f11&gt; S-&lt;right&gt;</b></li> <li><b>&lt;Esc&gt; S-&lt;right&gt;</b></li> </ul>	<b>(fm-right-frame)</b>	Move point to frame at right of the current frame (if one exists and is located there).
<b>Move to graphical frame at left</b>	<ul style="list-style-type: none"> <li><b>&lt;f11&gt; S-&lt;left&gt;</b></li> <li><b>&lt;Esc&gt; S-&lt;left&gt;</b></li> </ul>	<b>(fm-left-frame)</b>	Move point to frame at left of the current frame (if one exists and is located there).
<b>Manage Frames</b>	Emacs frame operations work in <b>both</b> modes: graphics and text terminal. In graphics mode a new frame is creating a separate OS frame. <ul style="list-style-type: none"> <li>In text terminal mode a new frame is inside the <b>same</b> OS frame: it simply hides the other(s) existing frame(s). The mode line identifies the currently active frame number as “F1”, “F2”, etc... The windows numbers are all part of the space number-space.</li> </ul>		
<b>Show frame count</b>	<b>&lt;f11&gt; F ?</b>	<b>(pel-show-frame-count)</b>	Display the number of Emacs active frames in the mini-buffer.
<b>Delete this frame</b>  See <a href="#">↶ Windows Hydra</a>	<ul style="list-style-type: none"> <li><b>C-x 5 0</b></li> <li><b>&lt;f11&gt; F 0</b></li> <li><b>⌘-w</b></li> <li><b>* &lt;f7&gt; M-F</b></li> </ul>	<b>(delete-frame</b> &optional FRAME FORCE)	Delete FRAME, permanently eliminating it from use. <ul style="list-style-type: none"> <li>FRAME must be a live frame and defaults to the selected one.</li> </ul>
<b>Delete all other frames</b>	<ul style="list-style-type: none"> <li><b>C-x 5 1</b></li> <li><b>&lt;f11&gt; F 1</b></li> </ul>	<b>(delete-other-frames</b> &optional FRAME)	Delete all frames on FRAME’s terminal, except FRAME.
<b>New Frame below</b>  See <a href="#">↶ Windows Hydra</a>	<ul style="list-style-type: none"> <li><b>C-x 5 2</b></li> <li><b>&lt;f11&gt; F 2</b></li> <li><b>⌘-n</b></li> <li><b>* &lt;f7&gt; M-f</b></li> </ul>	<ul style="list-style-type: none"> <li><b>(make-frame-command)</b></li> <li><b>(make-frame</b> &amp;optional PARAMETERS)</li> </ul>	Make a new frame, on the same terminal as the selected frame. <ul style="list-style-type: none"> <li>🍏 On macOS in graphics mode only: make-frame is called for <b>⌘-n</b>, it has the same effect as make-frame-command.</li> </ul>
<b>Display buffer in other (next) frame</b>	<ul style="list-style-type: none"> <li><b>C-x 5 C-o</b></li> <li><b>&lt;f11&gt; F b</b></li> </ul>	<b>(display-buffer-other-frame</b> BUFFER)	Display a buffer preferably in another frame.
<b>Tear window in frame</b> See <a href="#">↶ Windows Hydra</a>	<b>C-x w ^ f</b> <ul style="list-style-type: none"> <li><b>&lt;f11&gt; w i f</b></li> <li><b>* &lt;f7&gt; F</b></li> </ul>	<b>(tear-off-window</b> CLICK)	Delete the selected window, and create a new frame displaying its buffer. <ul style="list-style-type: none"> <li>See: <a href="#">↶ Windows</a></li> </ul>
<b>Select another frame</b>	<ul style="list-style-type: none"> <li>In text terminal mode: iterate through all frames, make the next frame visible in the terminal window and update the frame number shown on the mode line.</li> <li>In graphics mode: with no/nil argument (the default): iterate through visible frames, with ARG non nil: iterate through all frames (included iconified frames).</li> </ul>		
<b>Select next frame</b> • the frame with the next higher frame number	<ul style="list-style-type: none"> <li><b>&lt;f11&gt; F n</b></li> <li><b>⌘-`</b></li> <li><b>* &lt;f7&gt; }</b></li> </ul>	<b>(pel-next-frame</b> ARG)	Make next frame visible. <ul style="list-style-type: none"> <li>🍏 The <b>⌘-`</b> key is a macOSs command that moves to the next frame of the same application: no Emacs code is invoked but the effect is the same in graphics mode.</li> </ul>
<b>Select previous frame</b> • the frame with the next lower frame number	<ul style="list-style-type: none"> <li><b>&lt;f11&gt; F p</b></li> <li><b>⌘--</b></li> <li><b>* &lt;f7&gt; {</b></li> </ul>	<ul style="list-style-type: none"> <li><b>(pel-previous-frame</b> ARG)</li> <li><b>(ns-prev-frame)</b></li> </ul>	Make previous frame visible. <ul style="list-style-type: none"> <li>🍏 The <b>⌘-`</b> key is a macOSs command that moves to the next frame of the same application: no Emacs code is invoked but the effect is the same in graphics mode.</li> </ul>
<b>Move cursor to other (next) frame</b>	<ul style="list-style-type: none"> <li><b>C-x 5 o</b></li> <li><b>&lt;f11&gt; F o</b></li> </ul>	<b>(other-frame</b> ARG)	Activate the other frame. Numeric argument identifies frame in Z order.
<b>Open in frame</b>	The following commands open a buffer, file or <a href="#">↶ Dired</a> buffer in a frame.		
<b>Select buffer in other frame</b>	<ul style="list-style-type: none"> <li><b>C-x 5 0</b></li> <li><b>&lt;f11&gt; F 0</b></li> </ul>	<b>(switch-to-buffer-other-frame</b> BUFFER-OR-NAME &optional NORECORD)	Prompt for buffer and open in other frame.
<b>Find file in other (next) frame in read-only</b>	<ul style="list-style-type: none"> <li><b>C-x 5 r</b></li> <li><b>&lt;f11&gt; F r</b></li> </ul>	<b>(find-file-read-only-other-frame</b> FILENAME &optional WILDCARDS)	Edit file read-only in other frame with name obtained via minibuffer.
<b>Find file in other (next) frame</b>	<ul style="list-style-type: none"> <li><b>C-x 5 f</b></li> <li><b>C-x 5 C-f</b></li> <li><b>&lt;f11&gt; F f</b></li> </ul>	<b>(find-file-other-frame</b> FILENAME &optional WILDCARDS)	Switch to/open another file and show it in another frame.
<b>Run Dired in other (next) frame</b>	<ul style="list-style-type: none"> <li><b>C-x 5 d</b></li> <li><b>&lt;f11&gt; F d</b></li> </ul>	<b>(dired-other-frame</b> DIRNAME &optional SWITCHES)	"Edit" a directory. Like ‘dired’ but makes a new frame. <ul style="list-style-type: none"> <li>See <a href="#">↶ Dired</a></li> </ul>
<b>macOS frame</b>	The following commands are available under macOS for Emacs running in graphical mode only.		
<b>Hide Emacs frame</b>	<b>⌘-h</b>	<b>(ns-do-hide-emacs)</b>	🍏 On macOS in graphics mode only: hide Emacs frame. <ul style="list-style-type: none"> <li>Retrieve it via the <b>⌘-&lt;tab&gt;</b> key.</li> </ul>
<b>Hide all other applications</b>	<b>⌘-H</b>	<b>(ns-do-hide-others)</b>	🍏 On macOS in graphics mode only: hide all other applications, except Emacs. <ul style="list-style-type: none"> <li>Retrieve them via the <b>⌘-&lt;tab&gt;</b> key.</li> </ul>
<b>Iconify frame</b>	<b>⌘-m</b>	<b>(iconify-frame</b> &optional FRAME)	🍏 On macOS in graphics mode only: iconify the frame.