spmenu documentation 0.4.1

Code documentation

Some of spmenu's code documented. If you want to hack on the project, this should be useful. Note that these may be renamed in the future to make the codebase easier to understand and make changes to. Also note that this is **definitely not** a complete list.

Position and width/height variables

- `bh`
    - Menu height divided by lines gets you `bh`. The name comes from dwm's `bh` meaning 'bar height'. This is the height of each item in the list.
- `mh`
    - Menu height (or height of the window)
    - Use `drw_resize()` and `XResizeWindow()` to adjust this.
- `mw`
    - Menu width (or width of the window)
    - Use `drw_resize()` and `XResizeWindow()` to adjust this.
    - `2 * borderwidth` is removed from `mw` (2* because we have two sides)
    - `2 * sp` is removed from `mw` (2* because we have two sides)

- `x`
  - X position, functions like `drw_text` use this.
  - If you set this in bar drawing functions, you must apply the same to `buttonpress()`, otherwise clicks will be offset.
- `y`
  - Y position, functions like `drw_text` use this.
  - If you set this in bar drawing functions, you must apply the same to `buttonpress()`, otherwise clicks will be offset.
- `ev->x`
  - X position where you clicked. This is used in the `buttonpress()` function to check where you clicked.
- `ev->y`
  - Y position where you clicked. This is used in the `buttonpress()` function to check where you clicked.
- `w`
  - Width of something, this is passed to `drw_text()` for example, but you may override this.
- `plw`
  - This is the width of the powerline arrow. It must be added on in the `buttonpress()` and draw functions.
- `vp`
  - Vertical padding, this is initially added on in the `create_window()` function.
- `sp`
  - Horizontal padding, this is initially added on in the `create_window()` function.
- `promptw`
  - Width of the prompt text, this is going to be the same as `TEXTW(prompt)`.
- `inputw`
  - Width of the input text.
- `fh`
  - Font height. Used to calculate the height of the cursor. See `drawcaret()`.
- `menuposition`
  - Integer the user is meant to configure. If it’s set to `0`, spmenu will be put on the bottom of the screen. If it’s set to `1` it will be put on the top of the screen. If it’s `2` it will be put in the center of the screen.
- `wa.width`
  - Window width, `wa` is `XWindowAttributes`.
- `wa.height`
  - Window height, `wa` is `XWindowAttributes`.

- `imageheight`
  - Image height, This is **not** the height of the image, it is the height that the image will be scaled to fit.
- `imagewidth`
  - Image width, This is **not** the width of the image, it is the width that the image will be scaled to fit.
- `imagegaps`
  - Image gaps, this is extra space added around the image.
- `imageh`
  - Usually the same as `imageheight`. This is what `imageheight` is initially set to.
- `imagew`
  - Usually the same as `imagewidth`. This is what `imagewidth` is initially set to.
- `imageg`
  - Usually the same as `imagegaps`. This is what `imagegaps` is initially set to.
- `longestedge`
  - As the name implies, it is the longest (highest value) of `imageheight` and `imagewidth`.
- `numberWidth`
  - Integer set in some functions, it is simply `TEXTW(numbers)` if the match count isn't hidden.
- `modeWidth`
  - Integer set in some functions, it is simply `TEXTW(modetext)` if the mode indicator isn't hidden.
- `larrowWidth`
  - Integer set in some functions, it is simply `TEXTW(leftarrow)` if the left arrow isn't hidden.
- `rarrowWidth`
  - Integer set in some functions, it is simply `TEXTW(rightarrow)` if the right arrow isn't hidden.
- `powerlinewidth`
  - Integer set in some functions, it is simply `plw / 2` if powerlines are enabled.
- `curpos`
  - Cursor/caret position. When text is added to the input, the width of that text is added to this.

Drawable abstraction functions

Most of these are in `libs/sl/draw.c` and `libs/sl/draw.h`.

- `drw_create(Display *dpy, int screen, Window win, unsigned int w, unsigned int h, Visual *visual, unsigned int depth, Colormap cmap);`
  - This function creates a drawable from `Display *dpy`, `Drw`. Think of it as a canvas.
- `drw_resize(Drw *drw, unsigned int w, unsigned int h)`
  - This function resizes the drawable to the dimensions passed as arguments (`w`, `h`).
- `drw_free(Drw *drw);`
  - This function will free the drawable from memory. It is *usually* called in cleanup functions like `cleanup()` so most of the time you don't need to use this.

Font abstraction functions

Most of these are in `libs/sl/draw.c` and `libs/sl/draw.h`. NOTE: These will differ slightly depending on if Pango is enabled or not.

- `drw_font_create(Drw* drw, char *font[], size_t fontcount);`
  - This function will return a font libXft can use.
- `drw_font_free(Fnt *set);`
  - This function will free the font from memory.
- `drw_fontset_getwidth_clamp(Drw *drw, const char *text, unsigned int n, Bool markup);`
  - This function returns the smallest value out of the passed argument `n` and the length of the text drawn. The text is not actually drawn though.
- `drw_font_getwidth(Drw *drw, const char *text, Bool markup);`
  - This function returns the width of drawn text. The text is not actually drawn though.
- `drw_font_getexts(Fnt *font, const char *text, unsigned int len, unsigned   int *w, unsigned int *h, Bool markup);`
  - This function returns the length of the text with the used font.

Colorscheme abstraction functions

- `drw_clr_create(Drw *drw, Clr *dest, char *clrname, unsigned int alpha);`
  - This function allocates space for a color.
- `drw_scm_create(Drw *drw, char *clrnames[], unsigned int alphas[], size_t clrcount);`
  - This function returns a color scheme from an array of colors and alpha.

Cursor abstraction functions

- `drw_cur_create(Drw *drw, int shape);`
  - This function creates and returns a cursor.
- `drw_cur_free(Drw *drw, Cur *cursor);`
  - This function will free the cursor from memory.

Drawable context functions

- `~~drw_setfont(Drw *drw, Fnt *set);`
  - Sets the font.
  - NOTE: Applies only if Pango is disabled.~~
  - Removed, no longer necessary.
- `drw_setscheme(Drw *drw, Clr *scm);`
  - Sets the color scheme to `*scm` created by `drw_scm_create()`

Drawing functions

- `drw_rect(Drw *drw, int x, int y, unsigned int w, unsigned int h, int filled , int invert);`
  - Draws a simple rectangle. Used in other functions to create more useful shapes, such as a cursor.
- `drw_text(Drw *drw, int x, int y, unsigned int w, unsigned int h, unsigned int lpad, const char *text, int invert, Bool markup);`
  - Draws text on the drawable using the font created. `const char *text` contains the text itself.

Map functions

- `drw_map(Drw *drw, Window win, int x, int y, unsigned int w, unsigned int h);`
    - Maps the drawable. (makes it visible)