Evas Textblock Introduction

dan 'dj2' sinclair, The Enlightenment Project
<zero@everburning.com>

Table of Contents

Evas Textblock Introduction

First there was Etox, a handy little text layout library, but as more stuff got built on top of it a few cracks started to appear. Bugs popped up and it got complicated. So, out of the ether, Raster pulled the *evas_object_textblock* and slowly but surely we've been building up a sweet little text layout object.

There is still a bit of functionality to add to the textblock, including: anchors, overflow objects, obstacles and a few others. But, in terms of text handling, the textblock should do everything you need, and maybe more.

This article is broken into two parts. The first is an introduction to the textblock with a little example program. It doesn't do much of anything useful but should get you on your feet. The second section is a list of the different formatting tags that the textblock will accept and the parameters you can pass to each.

So, lets get to it. I'm going to give the entire program as one big ass chunk, then explain the smaller pieces that are relevant to the textblock. So, without further ado, here's some Shakespeare.

Example 1. Evas Textblock Example

```
#include <Ecore.h>
#include <Ecore_Evas.h>
#include <Evas.h>

#define WIDTH 300
#define HEIGHT 400

/* random Shakespeare quotes strung together */
char *long_text =  "To be or not to be, --that is the question:-- "
    "Whether 'tis nobler in the mind to suffer "
    "The slings and arrows of outrageous fortune "
    "Or to take arms against a sea of troubles, "
    "And by opposing end them? "
    "His life was gentle, and the elements so mixed in him that Nature might "
    "stand up and say to all the world, this was a man "
    "Good night, good night ! parting is such sweet sorrow, that I shall say "
    "good night till it be morrow "
    "To thee I do commend my watchful soul, ere I let fall the windows of "
    "mine eyes; sleeping and waking, O, defend me still "
    "What a piece of work is man!  how noble in reason! how infinite in "
    "faculty! in form and moving how express and admirable!  in action how "
    "like an angel! in apprehension how like a god! the beauty of the world, "
    "the paragon of animals!";

int
main (int argc, char ** argv)
{
    Ecore_Evas *ee;
    Evas *evas;
    Evas_Object *o;
    int ret = 1;

    if (!ecore_init())
    {
        printf("Unable to init ecore\n");
        goto EXIT;
    }
    ecore_app_args_set(argc, (const char **) argv);

    if (!ecore_evas_init())
    {
        printf("Unable to init ecore_evas\n");
        goto ECORE_SHUTDOWN;
    }

    /* create the ecore evas to display the textblock in */
    ee = ecore_evas_software_x11_new(NULL, 0, 0, 0, WIDTH, HEIGHT);
    ecore_evas_title_set(ee, "Textblock tests");
    ecore_evas_show(ee);

    evas = ecore_evas_get(ee);

    /* create the white bg image */
    o = evas_object_rectangle_add(evas);
    evas_object_move(o, 0, 0);
    evas_object_resize(o, WIDTH, HEIGHT);
    evas_object_color_set(o, 255, 255, 255, 255);
    evas_object_layer_set(o, -100);
    evas_object_show(o);
```

```
    /* create the textblock */
    o = evas_object_textblock_add(evas);
    evas_object_move(o, 0, 0);
    evas_object_resize(o, WIDTH, HEIGHT);
    evas_object_show(o);

    evas_object_textblock_format_insert(o, "color=#000000ff "
                "font=/usr/local/share/evas/Vera.ttf size=10");
    evas_object_textblock_text_insert(o, "Stick some text in here");
    evas_object_textblock_format_insert(o, "\n");
    evas_object_textblock_format_insert(o, "size=20");
    evas_object_textblock_text_insert(o, "Go big or go home");
    evas_object_textblock_format_insert(o, "size=10 color=#ff0000ff");
    evas_object_textblock_text_insert(o, "I like my home");
    evas_object_textblock_format_insert(o, "style=shadow align=center "
                "wrap=word color=#0000ffff left_margin=15% right_margin=-15%");
    evas_object_textblock_text_insert(o, long_text);

    ecore_main_loop_begin();
    ret = 0;

ECORE_EVAS_SHUTDOWN:
    ecore_evas_shutdown();
ECORE_SHUTDOWN:
    ecore_shutdown();
EXIT:
    return ret;
}
```

I'm going to skip the startup/shutdown code as you can learn about that in other places, I'm just going to focus on the textblock code itself. And yes, I know, goto is evil. Well, too damn bad. You can do it some other way in your code, but I think it makes a limited number of cases nicer and easier to follow.

Example 2. Creating the textblock

```
    /* create the textblock */
    o = evas_object_textblock_add(evas);
    evas_object_move(o, 0, 0);
    evas_object_resize(o, WIDTH, HEIGHT);
    evas_object_show(o);
```

Creating a textblock is the same as any other evas object, a simple call to `evas_object_textblock_add(Evas *evas)`. I then position the object to be the full size of the window.

If you want to set your textblock to be just big enough for the data inside it, the best way to do it is, after setting your text into the textblock, make a call to `evas_object_textblock_native_size_get(Evas_Object *obj, Evas_Coord *w, Evas_Coord *h)`. This will give you the width and height that the text would need in order to be layed out inside the textblock. This does not include formatting information. You can then resize your textblock to the given width and height.

There is also the `evas_object_textblock_format_size_get(Evas_Object *obj, Evas_Coord *w, Evas_Coord *h)` call, which will tell you the size of the textblock after

formatting. Just note, if you haven't set a width on your textblock this will return 0. This is because, if there is size for your textblock it can't format anything, so you get size 0 returned. You'll need to set an initial width/height on the textblock before trying to get the formatted size.

Example 3. Text and Formatting

```
evas_object_textblock_format_insert(o, "color=#000000ff "
            "font=/usr/local/share/evas/Vera.ttf size=10");
evas_object_textblock_text_insert(o, "Stick some text in here");
evas_object_textblock_format_insert(o, "\n");
evas_object_textblock_format_insert(o, "size=20");
evas_object_textblock_text_insert(o, "Go big or go home");
evas_object_textblock_format_insert(o, "size=10 color=#ff0000ff");
evas_object_textblock_text_insert(o, "I like my home");
evas_object_textblock_format_insert(o, "style=shadow align=center "
            "wrap=word color=#0000ffff left_margin=15% right_margin=-15%");
evas_object_textblock_text_insert(o, long_text);
```

Now that the textblock is created we can start sticking some text and formatting into it. (You can actually do this at any time after creating the textblock). The formatting used by the textblock is kept separate from the actual text itself. Formatting is inserted by calling `evas_object_textblock_format_insert(Evas_Object *obj, const char *format)` while text is inserted by calling `evas_object_textblock_text_insert(Evas_Object *obj, const char *text)`.

It should be noted that *\n* is a *formatting* node and not a text node. If this is in your text there is a good chance you'll get 'squares' in your output. You'll notice I called `evas_object_textblock_format_insert(o, "\n")` when I wanted to put a new line into the output.

The formatting for the text is a simple string in which you give it a list of commands, for example *color=#000000ff size=10* sets the colour to black with the alpha channel set to full and the font size to 10.

With the textblock most settings default to off. There is no default font or font size. By default underline, glow, and other styles won't show up until they have their colour set. So make sure you set any needed default values on the format block.

Text and formatting are inserted into the textblock at the current cursor position. Information about the cursor can be manipulated with calls to: `int evas_object_textblock_cursor_pos_get(Evas_Object *obj)` and `evas_object_textblock_cursor_pos_set(Evas_Object *obj, int pos)`.

If you need information about the text itself there are several calls to help you. `evas_object_textblock_text_insert(Evas_Object *obj, const char *text)` as we have already seen will insert the given text into the textblock at the current cursor position. `char *evas_object_textblock_text_get(Evas_Object *obj, int len)` will get *len* characters of text starting at the current cursor position, and to go along with that `evas_object_textblock_text_del(Evas_Object *obj, int len)` will delete *len* characters from the textblock at the current cursor position.

There are quite a few other functions to get the character at given positions, to get formatting information or to get line information. To learn about these functions take a look at the Evas header file and the source code (under *evas/src/lib/canvas/evas_object_textblock.c*).

Example 4. Compilation

```
zero@oberon [test] -> gcc -o text_intro `ecore-config --cflags --libs` \
    `evas-config --cflags --libs` text_intro.c
```

Compilation is easy enough, just grab the ecore-config and the evas-config cflags and libs, and your good to go.

If you run

```
text_intro
```

you should see a window with a bunch of text in it. If you just see a white window, check the font path in the *font=* format tag above and that the *Vera.ttf* exists at that location.

That's it. With this you should be able to go give evas_object_textblock a good poking on your own. So, what are you waiting for, go play.

Textblock Formatting Options

There are several options available to style your text through the textblock. Each entry is of the format of a *key=value* pair. Below is a list of the different keys and their applicable values. As an aside, you do *not* need quotes around the values.

It should be noted that formats affect text inserted *after* the format node is put in place. They will not change the current text.

source

If your pulling your font from somewhere other then a font file, like an EET or Edje for example, you will need to set the source to the source file. If your doing this then remember, the font you give later needs to have the full key name from the file. So if your using an Edje, for example, you would have a format of *font=fonts/Vera* as Edje puts the fonts under the *font/* key automatically. Your other option is to do `evas_font_path_append(evas, "fonts/")` then you won't have to worry about it.

Eg. *source=/path/to/file/default.eet*

font

This is the font to use. You can either give a key inside an EET/Edje file or the path to a font on the filesystem. The font path used can either be relative to an entry in your font path (as set with `evas_font_path_append`) or an absolute path.

Eg *font=/path/to/fonts/Vera.ttf*.

size

Sets the font size.

Eg *size=10*.

align

This will set the horizontal alignment of the text inside the format block. The value is one of:

- left
- middle
- center
- right

Eg. *align=middle*.

valign

This will set the vertical alignment of the text inside the format block. The value is one of:

- top
- middle
- center
- bottom
- baseline
- base

Eg. *valign=bottom*

color

This will set the colour of the text that is inserted. The value of the colour string can be in one of the following formats:

- #RRGGBB
- #RRGGBBAA
- #RGB
- #RGBA

Eg. *color=#000000ff*

underline_color

Set the colour of the underline used in this formated section. This accepts colours in the same formats as the *color* tag above.

E.g *underline_color=#f00*

double_underline_color

Set the colour of the double underline used in this formated section. This accepts colours in the same formats as the *color* tag above.

E.g *double_underline_color=#00ff00*

outline_color

Set the colour of the outline used in this formated section. This accepts colours in the same formats as

the *color* tag above.

E.g *outline_color=#ff0f*

shadow_color

Set the colour of the shadow used in this formated section. This accepts colours in the same formats as the *color* tag above.

E.g *shadow_color=#c0c0c0c0*

glow_color

Set the colour of the glow used in this formated section. This accepts colours in the same formats as the *color* tag above.

E.g *glow_color=#0ff*

outer_glow_color

Set the colour of the outer glow used in this formated section. This accepts colours in the same formats as the *color* tag above.

E.g *outer_glow_color=#000000ff*

backing_color

Set the colour of the backing used in this formated section. This accepts colours in the same formats as the *color* tag above.

E.g *backing_color=#ff0000*

strikethrough_color

Set the colour of the strikethrough used in this formated section. This accepts colours in the same formats as the *color* tag above.

E.g *strikethrough_color=#fff*

wrap

This will allow you to change the type of wrapping used in this formatted block. The value can either be *word* or something else. If *word* is given, the text will be wrapped on word boundaries, otherwise the text will be wrapped at character boundaries.

E.g *wrap=word*

underline

The underline option allows you to set the type of underline in this formatted block. This can be set to one of:

- off
- on

- double

E.g *underline=double*

strikethrough

The strikethrough key allows you to turn strikethrough on or off for the current format block. The value is one of:

- off
- on

E.g *strikethrough=on*

backing

The backing key allows you to set the backing option. The value is one of:

- off
- on

E.g *backing=on*

style

If you want to add a bit of extra flair to your text, the textblock has the ability to add *styles* to the text. These styles allow you to do things like add soft shadows or glow to the text. Remember, you will need to set the colour for the effect, *glow_color=#ff0000ff* for example, before you can use it in a style. The available styles are:

- off
- none
- plain
- shadow
- outline
- outline_shadow
- outline_soft_shadow
- glow
- far_shadow
- soft_shadow
- far_soft_shadow

E.g *style=soft_shadow*

left_margin/right_margin

If you need to adjust the margins of your textblock this can be done with the *left_margin* and *right_margin* formatting options. They both take values of the format:

off Use default margins

#% set the margin to a percentage of the width. E.g *left_margin=15%*

set the margin to the given position. E.g *right_margin=10*

E.g *left_margin=15%* or *right_margin=-15%*

left_tab_stop/right_tab_stop

If you need to set the tab stop of the textblock then these are your keys. The different ways to specify the values are given below:

88 jump to X pos 88 from the left

47% jump to X pos that is 47% of the width from the left

+91 add 91 to X pos

-12 subtract 12 from X pos

+16% add 16% of width to X pos

-25% subtract 25% of width from X pos

E.g *left_tab_stop=88* or *right_tab_stop=-25%*

End matter

You should now have a good idea of the different format options and the values to go along with them. If you have any questions/comments or suggestions feel free to send them my way at zero@everburning.com.