Tekgui.a

Version: 1.0
By : Yan Poinssot

1. Introduction

Tekgui.a is a graphical library relying upon the liblapin library. Thanks to this library, creating and implementing graphical components has never been easier. This library offers you a great deal of customization to fuflfil all your needs without compromising graphical design. This documentation will allow you to explore all the possible hidden gems this library has to offer.
Liblapin doc: https://liblapin.readthedocs.org/fr/latest/

2. Tutorials

a. INI format:

i. Component types:

1. checkbox : A box which can be checked or unchecked.
2. imagebox: A compon ent which renders an image or part of it onto the graphical interface.
3. textbox: A graphical component which accepts text.
4. button: A component that can be pressed to trigger a defined action.

ii. Component fields:

1. *type : indicates the component to be rendered on the interface.
2. *name : name of the component *(**Each component** needs to have a **different name**)*.
3. *dimension: accepts an x an y value to then draw a component accordingly.
4. *position: accepts an x an y value to then position a component accordingly.
5. border : Defines whether the component is surrounded by a border *(default : false)*.
6. text : Defines the text written inside a certain component *(**Doesn't apply** to **checkboxes**)*.
7. check : Field that defines whether or not the checkbox is checked *(default : not selected, **Only** applies to **checkboxes**)*.
8. img : Field that accepts an address to an image *(only applies to **imagebox**)*.

9. function : Field that accepts the name of a function as a first parameter, and that accepts its argument as a second parameter. It will be triggered when the user interacts with the component.

iii. INI syntax:

The INI filetype is what's used to load graphical components with their properties.

1. Scope : the scope is what indicates to the ini parser the name of the item.
   The scope has to start by "item" and then be followed by the item number.
   If two items have the same scope, only the first one will be taken into account.
   If you skip a number the parser will not read any of the data after the skipped number.
2. Field : to enter a field with a specific value: "field_name:value1,value2".
   To separate the fields, add a comma inbetween.
   Do not add any spaces before or after the field name or the arguments' value, unless you want the space to be part of the value you are entering.
   If you add too many parameters for a certain field, the excess parameters won't be taken into account.

```
1  [item1]
2  type=button
3  name=button1
4  color=yellow
5  position=20,20
6  dimension=80,30
7  border=false
8  function=hello
9  check=1
10
11 [item2]
12 type=checkbox
13 name=button2
14 color=red
15 text=Nice!
16 position=20,60
17 dimension=30,30
18 border=true
19 check=1
```

The default scope (no scope given) contains two important components:
max_ram : *max amount of ram used*
font : *Directory of the font used by the program.*

iv. INI example:

Item1: button called button1, yellow, positioned at x = 20 and y = 20, with a dimension of x = 80 and y = 30 with a border and Hello written on it.
Item2: checkbox called choose, yellow, positioned at x = 110 and y = 20, with a dimension of x = 30 and y = 30 with a border.

Item3: textbox called Hello world, orange, positioned at x = 20 and y = 60, with a dimension of x = 120 and y = 200 with a border.

b. Implementation of the library:

i. Compilation:

To use the content of the library tekgui.a you need to compile your files with the tekgui library and the liblapin. If you compile in the terminal add all the following libraries **on the same line**:

```
gcc file.c lib.a
-rdynamic
-I/home/${USER}/.froot/include
-L/home/${USER}/.froot/lib -llapin -lsfml-audio
-lsfml-graphics -lsfml-window -lsfml-system
-lstdc++ -ldl -lm
```

if you are compiling with a Makefile, refer to the one provided in "./resource".

ii. Includes:

At the start of all your files, include the header file of the tekgui library and the liblapin:

```
#include   <tekgui.h>
 #include   <lapin.h>
```

iii. Example of Implementation : *"./resource/main.c"*

```c
#include <lapin.h>
#include <unistd.h>
#include "./../include/tekgui.h"

t_bunny_response        loop(void                       *import)
{
  t_win                 *data;
  t_bunny_position      pos;

  data = (t_win *)(import);
  pos.x = 0;
  pos.y = 0;
  tekgui_display(data->pix, data->gui);
  bunny_blit(&(data->win)->buffer, &(data->pix->clipable), &pos);
  bunny_display(data->win);
  return (GO_ON);
}

int                     main(int                        argc,
                             char                       **argv)
{
  t_win                 data;
  t_bunny_pixelarray    *pix;
  t_bunny_window        *win;
  const char            *file;

  file = (const char *)(argv[1]);
  if (argc > 0 && (data.gui = tekgui_load(file)) == NULL)
    return (1);
  win = bunny_start(640, 480, false, "tek_gui");
  pix = bunny_new_pixelarray(640, 480);
  data.win = win;
  data.pix = pix;
  bunny_set_loop_main_function(&loop);
  bunny_loop(win, 30, &data);
  bunny_delete_clipable(&(pix->clipable));
  bunny_delete_clipable(&(data.gui->font->clipable));
  bunny_stop(win);
  return (0);
}
```

1. Loading an INI file into a t_tekgui*

   Before displaying a .ini file, you have to load it onto a `t_tekgui*` structure (for more information on the function, refer to the subchapter *Functions*):

   ```c
   const char            *file;

   file = (const char *)(FILENAME);
   if (data.gui = tekgui_load(file) == NULL)
     return (1);
   ```

   As the snippet above shows, change the field `FILENAME` with the filename of the ini you want to load. The function `tekgui_load` will return an item of type `t_tekgui*` or NULL if an error occured.

2. Display t_tekgui*:

   As shown on the example, a `t_win` structure was created. This structure is crucial for the program as it is responsible for:

   - Storing the info needed to display the interface.
   - Managing the mouse and keyboard events.

```
t_win                   data;
t_bunny_pixelarray      *pix;
t_bunny_window          *win;
const char              *file;

file = (const char *)(argv[1]);
if (argc > 0 && (data.gui = tekgui_load(file)) == NULL)
  return (1);
win = bunny_start(640, 480, false, "tek_gui");
pix = bunny_new_pixelarray(640, 480);
data.win = win;
data.pix = pix;
```

The structure contains a:

`t_bunny_window`, `t_pixelarray` and a `t_tekgui`.

As the snippet shows, the `t_tekgui` loaded has to be stored on the `gui` field, the `t_bunny_window` on the `win` field and the `t_bunny_pixelarray` on the `pix` field of the `t_win` structure.

```
bunny_set_loop_main_function(&loop);
bunny_loop(win, 30, &data);
bunny_delete_clipable(&(pix->clipable));
bunny_delete_clipable(&(data.gui->font->clipable));
bunny_stop(win);
return (0);
```

Then call the function `bunny_set_loop_main` function with `&loop` as an argument; this will tell to the liblapin which function you assign as the looping function when calling `bunny_loop`. Call the function `bunny_loop` right afterwards; the first argument indicates the `t_bunny_window` the library will modify, the second argument is the frequency of refresh of the window, finally, the third argument has to be the `t_win` structure you created previously.

```
t_bunny_response        loop(void                       *import)
{
  t_win                 *data;
  t_bunny_position      pos;

  data = (t_win *)(import);
  pos.x = 0;
  pos.y = 0;
  tekgui_display(data->pix, data->gui);
  bunny_blit(&(data->win)->buffer, &(data->pix->clipable), &pos);
  bunny_display(data->win);
  return (GO_ON);
}
```

The `void*` argument `import` is your `t_win` structure that you have to cast back into its own entity. Once you casted your structure, call the function `tekgui_display` with, as a first argument the `t_bunny_pixelarray` that will be modified, and as a second argument, the `t_tekgui` that will be read and modified according to the user. Finally

call the functions `bunny_blit` and `bunny_display` to copy the `t_bunny_pixelarray` onto `t_bunny_window`.

3. Characteristics

a. Structures:

i. t_item :

*Structure co*ntaining all the fields of a component.

char ***type** : *field containing the type of the component.*

char ***name** : *field containing the name of the component.*

char ***color** : *field containing the color of the component (only colors present inside the color code are taken into account, see in the subchapter Colors).*

char ***text** : *field containing text present inside the component (unused for checkboxes).*

char ***img** : *field containing the address of the picture to be loaded if the component is an imagebox.*

char ***border** : *field containing whether the component is surrounded by borders; if true, field is equal to "true".*

char **state** : *field determining if the component is being clicked on by the mouse; only one component can be selected in total; if true, status = '1' else status = '0' (doesn't apply to checkboxes).*

char **check** : *field only applying to checkboxes, determining if the component is crossed or not; only one component can be selected in total; if true, status = '1' else status = '0' .*

t_bunny_pixelarray ***pix** : *field only applying to imageboxes, it contains the pixelarray of the picture loaded according to the field img.*

t_bunny_position **pos** : field determining the x and y coordinates of a component.

t_bunny_position **dim** : field determining the x and y dimensions of a component.

(void) **(*function)(void *)** : field that defines the function that will be triggered during than event.

void ***argument** : field that defines the argument sent in the function.

ii. t_list :

Structure containing all the required information stored in the ini file, stored as a linked list.

t_item item : *Field containing the information for one graphical component.*

t_list *next : *Field containing the address of the next item within the linked list.*

iii. t_tekgui :

Main structure used by `tekgui_display` *to hold all the information needed to display the interface and interact with the user.*

int clicked : *Field used by the function* `mouse` *to interact with the user, defines if the mouse is clicking on a component.*

int max_ram : *Field set by default at 15 Mo which can be changed inside the INI file inside the default scope (max_ram=VALUE).*

t_list *list : *Field containing a list of all the components loaded by the* `tekgui_load` *function.*

char *font_adress : *Field containing the address of the font to be used by the interface, this setting is set inside the default scope of the INI file(font=ADRESS.png).*

t_bunny_pixelarray *font : *Field containing the font loaded inside the* `tekgui_load` *function according to the font_adress field.*

iv. t_win :

Structure used by `bunny_loop` *to hold all the fields needed to display the refreshed interface.*

t_tekgui *gui : *field containing all the information of the interface, used by the* `tekgui_dislay` *function.*

t_bunny_pixelarray *pix : *Field containing the pixelarray which holds all the modifications done by the* `tekgui_dislay` *function.*

t_bunny_window *win : *Field containing the window which will be displayed by the* `bunny_display` *function.*

b. Functions:

i. t_win :

t_tekgui *tekgui_load(const char *file);

Function that accepts an INI file directory and that extracts all the information required to create a t_tekgui structure according to the characteristics of the file.

void tekgui_display(t_bunny_pixelarray *pix, t_tekgui *gui);

Function that accepts a t_tekgui structure and that displays all of its info accordingly on the pixelarray pix.

void set_max_heap_size(size_t);

Function setting the maximum amount of ram used by the program.

void tekpixel (t_bunny_pixelarray *pix, t_bunny_position *pos, t_color *color);

Function that paints a pixel at the position pos with the color color.

void tekline (t_bunny_pixelarray *pix, t_bunny_position *pos, t_color *color)

Function that draws a line between the coordinates of pos[0] and pos[1] onto pix with the t_color provided.

void *tekfunction (cont char *str);

Function that returns the pointer of the function with the name defined in str.

void my_putstr (char *word);

Function that writes on the stdout the content of word.

void my_putstr_err (char *word);

Function that writes on the stderr the content of word.

char *my_malloc (int size);

Function that allocates memory for a char and that prints an error message in case of an allocation failure.*

char *my_strcpy (char *word);

*Function that copies the content of a char * on a new char * that it returns.*

int my_strcmp (char *word, char *words);

Function that compares 2 words and returns 0 if they are equal.

void add_let (char **text, char let);

Function that accepts the adress of a char * and adds let to its end.

char check_item (cont t_bunny_position *pos, t_list *list, int *pos);

*Function used to know if a position on a window is inside a component or not. It returns a letter according to the component which is crossed by the position ('a'=none, 'b'=button, 't'=textbox, 'c'=checkbox, 'i'=imagebox). At the same time, the int *pos stores the position of the item within the t_list.*

int set_selected (t_list **list, int pos, char state);

Function that sets a particular t_item->state according to the 2nd argument. At the same time, all the other t_items are set

to '0' since only one t_item can be selected at the time.

int set_checked (t_list **list, int pos);

Function that checks a checkbox if it is not checked or unchecks it if it is already checked. The int pos contains the position of the checkbox within the t_list.

t_item *getwrittenbox(t_tekgui *gui);

Function that returns the t_item (component has to be a textbox) which is selected.

void square(t_bunny_pixelarray *pix, t_bunny_position pos, t_bunny_position dim, t_color *color);

Function that draws a square, starting at the x,y coordinates with the dimensions of dim with the color defined by the last argument.

void square_line(t_bunny_pixelarray *pix, t_bunny_position pos, t_bunny_position dim, t_color *color);

Function that draws the borders of a square, starting at the x,y coordinates with the dimensions of dim with the color defined by the last argument.

void selected(t_bunny_pixelarray *pix, t_bunny_position pos, t_bunny_position dim, t_color *color);

Function that accepts the position and dimension of a component and that draws accordingly a selected icon on its bottom right onto the pix.

void selected(t_bunny_pixelarray *pix, t_bunny_position pos, t_bunny_position dim, t_color *color);

Function that accepts the position and dimensions of a checkbox and that crosses the box onto the pix..

void gradient(t_bunny_pixelarray *pix, t_bunny_position pos, t_bunny_position dim, t_color *color);

Function that accepts the position and dimensions of a square and the draws a gradient according to these informations on the pix.

unsigned int tekgetpixel(t_bunny_pixelarray *pix, int x, int y);

Function that returns the unsigned int value (color) of a pixel located at the x,y coordinates on the pixelarray.

void background (t_bunny_pixelarray *pix, t_color *color);

Function that paints the whole window according to the color given

void paste_pix (t_bunny_pixelarray *copy, t_bunny_pixelarray *out, t_bunny_position pos, t_bunny_position dim);

Function that copies the content of a the pixelarray copy (according to dim) onto the pixelarray out at the position pos.

char *pick_color()

Function that picks a different color from the color code each time it's called.

int change_color(char *name, t_list *list)

Function that changes the color of a given component (identified by the field "name").

4. Others

a. Colors:

Color	Background	Foreground
pink	11240191	9005261
purple	16724123	9116245
red	3881966	2302859
brown	1262987	997002
orange	36095	26317
yellow	55295	1940429
green	4033390	3107669
turquoise	13485312	9143808
blue	14772545	9125927