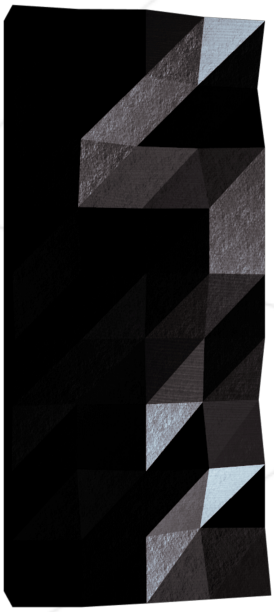

GUImp

Your first graphical GUI

Gadi gadi@42.fr
Jpeg jpeg@42.fr
Team Pedago pedago@42.fr

Summary: *The objective of this project will be to create your first graphical interface in order to create a reusable library in all your projects later !*

Contents

Chapter I

Foreword

On May 9, 1969, two kilometres from the border northern Laos, the 101st Infantry Division of the American army is advancing towards hill 937.

For them, it is a simple recognition. For the Vietcong, hill 937 is a strategic area.

The ten callers from the 101st Division, little trained in combat, was to carry out this routine mission in less than two hours.

They will resist heroically for nine days.

`This project does not tell their story.`

Figure I.1: GUI...mp

Chapter II

Introduction

A graphical interface - or more generally **GUI** *"Graphical User Interface"*,
is now a standard for any good software that respects itself, however this has not always been the case ...

At the beginning of the computer age, the only type of display that existed was the **CLI** *"Commande-line Interface"* with whom you have now more than familiarized yourself.

Since **CLI** was not created to make computing easy to use for ordinary people, it was for this very purpose that the graphical user interface was created in the 1970s by the engineers of the `Xerox Palo Alto Research Center`.
Launched for the first time with the `Star` and `Xerox`, it was greatly popularized with the commercialization of the `Macintosh` by `Apple` in 1984.

Figure II.1: CLI vs GUI.

Now it's up to you to create your own GUI library to create beautiful, easy-to-use software for the average person who doesn't know what a terminal is!

Chapter III

Objectives

The objective of this project will be for you to create a graphical interface library. This library should be as complete and modular as possible, the aim being to reuse it in your future projects, whether for your graphic branch projects or your other projects.

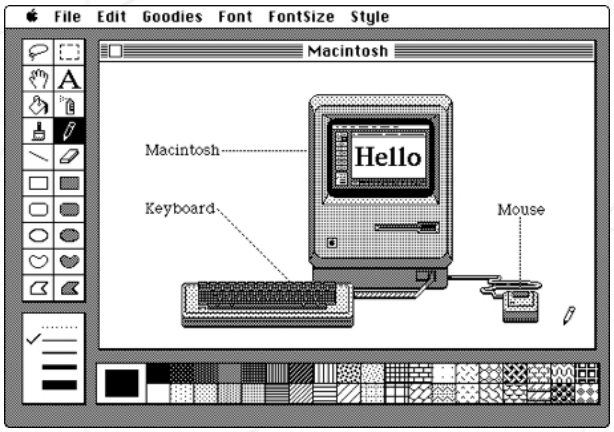

Figure III.1: The original interface of Paint - Macintosh in 1984.

To do this you will have to realize a small software of retouching and creation of digital images based on `The GIMP` to show all the functionalities as well as the good functioning of your library.

Chapter IV

General Instructions

- This project will be corrected by humans only. You're allowed to organise and name your files as you see fit, but you must follow the following rules.
- The executable file must be named `guimp`.
- Your `Makefile` must compile the project and must contain the usual rules. It must recompile and re-link the program only if necessary.
- If you are clever, you will use your `libft` for your `Doom-Nukem`. Submit also your folder `libft` including its own `Makefile` at the root of your repository. Your `Makefile` will have to compile the library, and then compile your project.
- You must render your library `libui` in a separate folder, at the root of your repository, named `libui`.
- You must submit a `Makefile`, containing the usual rules, that will compile a `libui.a`. This lib have to be link to your project like your `libft`.
- You cannot use global variables.
- Your project must be in `C` to the Norm.
- You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).
- Your program cannot have memory leaks.
- You'll have to submit a file called `author` containing your username followed by a '\n' at the root of your repository.

```
$>cat -e author
xlogin$
ylogin\$
```

- You must use for this project the `SDL2`. Either the version installed on clustered machines, or else directly via its sources in your repository, it should automatically

compile exactly like the `MinilibX` or your `libft`on your other projects, without any action other than compiling your rendering.

- As part of the mandatory part, you have the right to use the next libc function :
  - `open`
  - `read`
  - `write`
  - `close`
  - `malloc`
  - `free`
  - `perror`
  - `strerror`
  - `exit`
  - All functions of the math library (`-lm` and `man 3 math`)
  - All functions of `SDL 2`, `SDL_ttf` and `SDL_image`.
- You are allowed to use other functions or other librairies to complete the bonus part as long as their use is justified during your defense. Be smart!
- You can ask your questions on the forum, on slack...

Chapter V

Mandatory part

This mandatory part will be split into 2 parts :

- A library named `libui` containing all your Interface functions using the `SDL 2`.
- A drawing/retouching software similar to a `GIMP` or `Paint` using your `libui` and only this one for all your graphic management.

`Your reference guide for the SDL 2 https://wiki.libsdl.org/`

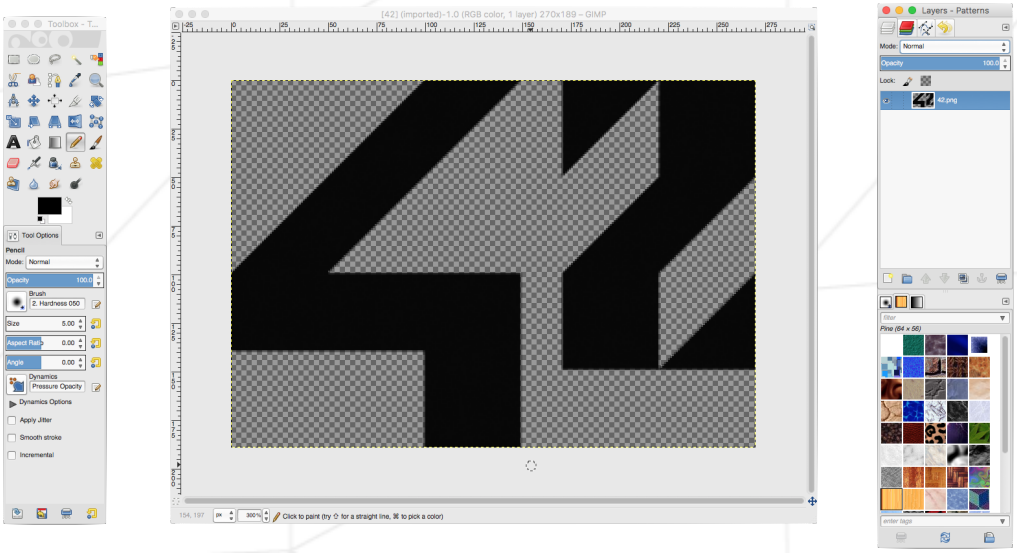

Figure V.1: Interface of The GIMP.

Figure V.2: Simple Directmedia Layer

V.1 Back: libui

You must create a library `libui` in the same way as your `libft` who will be able to :

- Create graphical windows on the screen, with different (size, resizeable, color and/or background image, particular effect, theme or style, etc...)
- Propose different types of windows: generic with elements, modal `OK`, modal `OK/CANCEL`..
- Generic windows can contain various elements: button, menu, text, image, editable text line and area, checkbox, radio, sliders, drop-down list, progress bar, etc...
- Each element has style or functionality parameters, e.g. a push button or on/off, colour and font for text, theme or style...
- Several elements may also contain elements, like a menu in a menu, an image in a button, etc... Several different "cascades " must be presented.
- Your `libui` manages user events associated with windows and elements. Click, focus, key, etc... it's possible for each element, if it makes sense, to specify a behaviour or special action related to each event. This can be a default action of the element (increment a counter for example), and thus managed by your libui, or an action defined by the user.
- The good cohabitation of the 2 previous points must be put into evidence.
- must be possible to have an interaction on an element which results in the appearance and/or modification of another element, e.g. a button set to ON shows an additional option.
- A system of `HotKeys` specific to each element type is available by default.
- A window can be scrollable if necessary according to the selected parameters.

- An element can be scrollable if applicable and necessary, according to the chosen parameters.
- Create some "préfabs" : ready-made combinations of elements compounds for common applications, such as a bar with images and info, or a default menu with open/save/quit etc..
- You must have a prefab of file selection and a prefab of choice of fonts available.
- Some elements and otherwise the window are capable of managing the drag-n-drop.
- All these features, if they are not used in the part will have to be demonstrated in defense.

Figure V.3: The Gimp

V.2 Front: Guimp

For the part `guimp` :

- Press the key `ESC` must close windows and exit the program properly.
- Click on the red cross on the border of the window must close the corresponding window and exit the program properly if this is the rendering window.
- Render image in a single window.
- PTool palette in a single window containing at least the following elements :
  - A button Brush
  - A button Eraser
  - Line drawing
  - A menu for drawing lines, rectangles, squares and circles (void `and` full)
  - A button Magnifying glass and moving hand
  - A line thickness management menu
  - A menu to import several images in the rendering
  - Colour selection to specific menu, circle RGB and/or sliders R-G-B
  - One button to clear the workspace
  - A brush menu with which we must be able to select and put "stickers" on the rendering and this in a `fluid way`
  - A button that fills a shape, aka the tool `painting pot`

  - A menu to display text at a given location, with choice of color, font and size
  - A pipette button, allowing you to select a color on an image
- Change the cursor according to the selected tool.
- Save the image in the format of your choice.
- Import of images in `JPEG` and `PNG`, but above all in **JPEG**..

`For all graphic management, your program will have to use your libui and this one only. Your program should therefore under no circumstances directly use the SDL, the system graphics framework or any other system graphic management.`

Chapter VI

Bonus part

Bonuses will be evaluated only if your mandatory part is PERFECT.
By PERFECT we - naturally - mean that it needs to be complete, that it cannot fail, even in cases of nasty mistakes like wrong uses etc.
Basically, it means that if your mandatory part does not obtain ALL the points during the grading, your bonuses will be entirely IGNORED.

- Undo/Redo (Cmd+z et Cmd+y)
- Copy and paste a selection
- Layer management
- Menus with tabs
- An interface style configuration file such as an file `CSS`.
- Internal drag and drop. (ex: layer gimp).
- Resizable interface..
- Responsive interface !
- Other things we never even imagined.

Chapter VII

Submission and peer correction

Submit your work on your `GiT` repository as usual. Only the work on your repository will be graded.

Good luck to all and don't forget your author file!