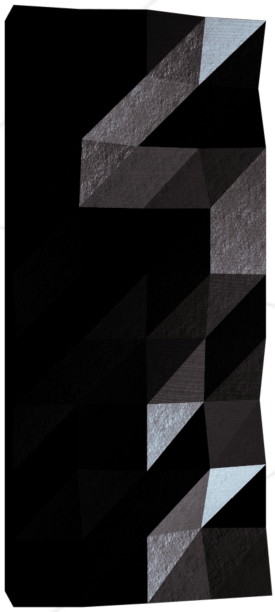

SoLong

And thanks for all the fish!

Summary: This project is a very small 2D game. It is built to make you work with textures, sprites. And some very basic gameplay elements.

Contents

Chapter I

Foreword

Being a developer is a great thing for creating your own game.
But a good game needs some good assets.
for 2D games, you should search for tiles, tilesets, sprites, and sprite sheets.
Thankfully some talented artists are willing to share their works on platforms like:
itch.io

> Whatever you do try to respect the works of others.

Chapter II

Goals

This project's objectives are similar to all this first year's objectives: Rigor, use of `C`, use of basic algorithms, information research etc.

As a graphic design project, `so long` will enable you to improve your skills in these areas: windows, colors, events, textures, etc.

Chapter III

Common Instructions

- Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check and you will receive a `0` if there is a norm error inside.
- Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a `0` during the evaluation.
- All heap allocated memory space must be properly freed when necessary. No leaks will be tolerated.
- If the subject requires it, you must submit a `Makefile` which will compile your source files to the required output with the flags `-Wall`, `-Wextra` and `-Werror`, and your Makefile must not relink.
- Your `Makefile` must at least contain the rules `$(NAME)`, `all`, `clean`, `fclean` and `re`.
- To turn in bonuses to your project, you must include a rule `bonus` to your Makefile, which will add all the various headers, librairies or functions that are forbidden on the main part of the project. Bonuses must be in a different file `_bonus.{c/h}`. Mandatory and bonus part evaluation is done separately.
- If your project allows you to use your `libft`, you must copy its sources and its associated `Makefile` in a `libft` folder with its associated Makefile. Your project's `Makefile` must compile the library by using its `Makefile`, then compile the project.
- We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded**. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Chapter IV

Mandatory part - so long

Program name	`so_long`
Turn in files	`All your files`
Makefile	`all, clean, fclean, re, bonus`
Arguments	`a map in format *.ber`
External functs.	• `open, close, read, write, printf, malloc, free, perror, strerror, exit` • `All functions of the MinilibX`
Libft authorized	`Yes`
Description	`You must create a small 2D game where a dolphin escapes earth after eating some fish. Or any hero collects any valuables before leaving the place.`

The constraints are as follows:

- You `must` use the `miniLibX`. Either the version that is available on the operating system, or from its sources. If you choose to work with the sources, you will need to apply the same rules for your `libft` as those written above in `Common Instructions` part.
- The management of your window must remain smooth: changing to another window, minimizing, etc.
- examples are given with a dolphin theme but you can use anything you want.

- The map will be constructed with 3 components walls, collectibles, and free space.
- The player's goal is to collect all collectibles present on the map then escape with minimal movement.
- At every move the current number of movements must be displayed in the shell.
- The player must be able to move: up, down, left, right.
- You will use a 2D view (top-down or profile).
- The game doesn't need to be real-time.
- player cannot move into walls.
- The program displays the image in a window and respects the following rules:
  - The W, A, S, and D keys will be used to move the main character.
  - Pressing `ESC` must close the window and quit the program cleanly.
  - Clicking on the red cross on the window's frame must close the window and quit the program cleanly.
  - The use of `images` of the `minilibX` is strongly recommended.
- Your program must take as a first argument a map description file with the `.ber` extension.
  - The map must be composed of only 5 possible characters: **0** for an empty space, **1** for a wall, **C** for a collectible, **E** for map exit and **P** for the player's starting position.
    This is a simple valid map:

```
1111111111111
100100000000C1
1000011111001
1P0011E000001
1111111111111
```

  - The map must be closed/surrounded by walls, if not the program must return an error.
  - Map must have at least one exit, one collectible, and one starting position.
  - You don't need to check if there's a valid path in the map.
  - The map must be rectangular.
  - You must be able to parse any kind of map, as long as it respects the rules of the map.

- Another minimal **.ber** map:

```
11111111111111111111111111111111
1E000000000000C00000C000000000001
10100101001000001010010000000010101
10100100101010100010010000000010101
1P0000000C00C000000000000000000C1
11111111111111111111111111111111
```

- If any misconfiguration of any kind is encountered in the file, the program must exit properly and return "Error\n" followed by an explicit error message of your choice.

Chapter V

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. It means that if your mandatory part does not obtain ALL the points during the grading, your bonuses will be entirely IGNORED.

Bonus list:

- enemy patrols that cause the player to lose in case of contact.
- There's some sprite animation.
- Movement count is directly displayed on the screen instead of shell output.

You will be able to create better games later do not waste too much time!

Chapter VI

Examples

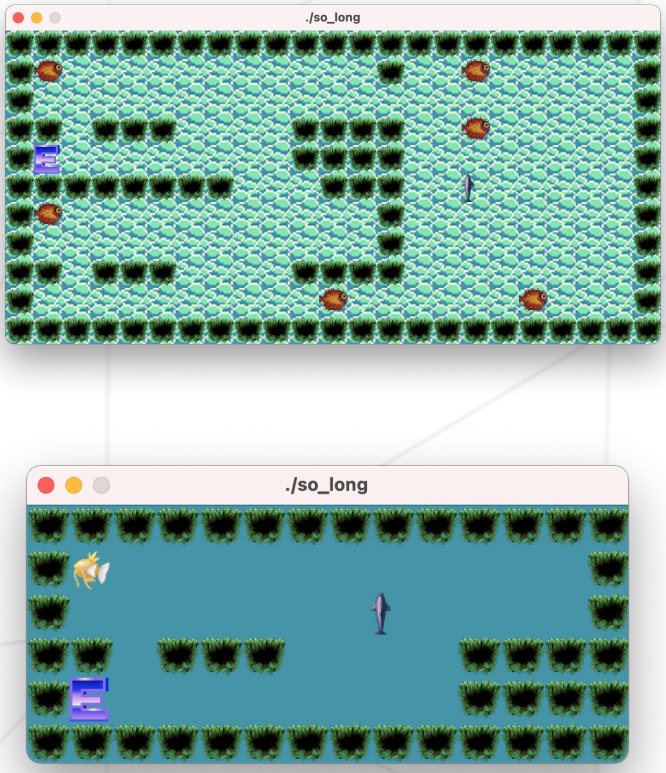

Figure VI.1: some `so_long` examples with some very bad (almost bonus worthy) taste in graphic design !