A

Contenido

[INTRODUCCIÓN 3](#_Toc129168101)

[INICIALIZACIÓN Y MAIN 4](#_Toc129168102)

[MAPA 5](#_Toc129168103)

# INTRODUCCIÓN

Esta es una pequeña guía donde explico los pasos que he ido siguiendo y cómo he hecho el proyecto de 42 de so\_long.

Para empezar, [aquí](https://qst0.github.io/ft_libgfx/man_mlx_new_image.html) hay una pequeña guía sobre varias funciones de la minilibx que son bastantes útiles y he utilizado. Aunque lo primero de todo sería explicar qué es la minilibx. De las 3 que se nos proporciona en la intra yo he utilizado es la minilibx\_opengl. Una minilibx es una biblioteca que proporciona una interfaz para realizar gráficos básicos en 2D, como dibujar líneas, círculos, rectángulos y mostrar imágenes, aunque en nuestro caso no utilizaremos tantas funciones, sino que la usaremos principalmente para el manejo de imágenes, eventos…

También hay que tener en cuenta que para poder ejecutar nuestro programa necesitamos enlazar las librerías necesarias con nuestro programa. Esto habrá que hacerlo más tarde en el makefile, pero hasta que lo tengamos acabado podemos compilar nuestro proyecto de esta forma:

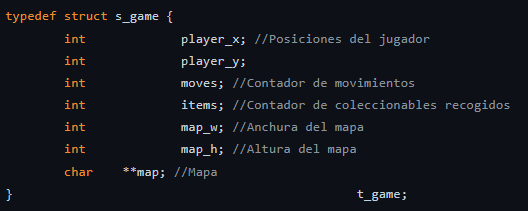
gcc -Wall -Wextra -Werror -Lmlx -lmlx -framework OpenGL -framework AppKit “tus archivos”

Para realizar el proyecto creo que lo mejor es empezar siguiendo esta [guía](https://harm-smits.github.io/42docs/libs/minilibx/getting_started.html) para aprender las funcionalidades básicas de la biblioteca, no hace falta seguirla al completo, lo importante es entender cómo funcionan las imágenes, las pantallas, los hooks… Una vez se comprendan todos estos conceptos será mucho más fácil entender bien cómo funciona el so\_long.

En esta guía no explicaré demasiado la construcción de un makefile (aunque si se verá un poco más adelante) ni tampoco paso por paso qué añadir a nuestro .h, ya que deberán ser las cabeceras, constantes, structs… Lo que vaya a utilizar cada uno.

Y tampoco iré poniendo todas las capturas de pantalla de todos los fragmentos de código, solamente iré poniendo los principales o cosas específicas. Para ver todo el código podéis ir a mi github.

# INICIALIZACIÓN Y MAIN

Para empezar, necesitaremos una variable donde guardar la información de nuestro juego, hay varias formas de hacerlo, pero la más sencilla es definirnos un par de structs. En uno guardaremos los punteros que nos referenciaran a la librería gráfica, a la ventana que tengamos abierta, a la imagen que estemos utilizando y al resto de información relevante. Esta última se puede declarar en otro struct donde guardaremos, por ejemplo, posiciones del jugador, número de movimientos, datos sobre el mapa…

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Estos structs los estaremos utilizando durante todo el proyecto, por lo que es muy importante saber qué hace cada uno de ellos.

Para hacer nuestro main, además de hacer la comprobación de errores correspondientes, la creación e inicialización de una variable t\_data y la asignación de memoria a su puntero game, también deberíamos: Dar unos valores iniciales a game, leer un mapa que se nos pase (Deberemos comprobar en un futuro si ese mapa es correcto),

¿QUÉ MAS?

# MAPA

Sobre el mapa vamos a tener que realizar bastantes operaciones. Para empezar deberemos comprobar que el mapa que se nos ha pasado es correcto, tanto el archivo como su composición. También deberemos extraer sus dimensiones, para añadirlas al puntero data y por último también deberemos “parsear” el mapa. Cuando “parseamos” el mapa lo que estamos haciendo es analizar sus cadenas de caracteres, para identificar cómo está hecho (paredes, obstáculos…).

## Comprobación del mapa

A grandes rasgos vamos a tener que comprobar solamente 2 cosas. La primera es el tipo de archivo, buscamos que se nos pase un .ber, y debemos comprobar que así sea. Y también debemos asegurarnos de que cumple los requisitos que se nos piden en un mapa, y esto último es bastante más complicado.

xd

## Parseo del mapa

¿?