Proyecto So_Long - Análisis Detallado del Código

Estructura del Código y Funcionalidad - Proyecto So_Long

1. Makefile - Explicación Línea por Línea

El Makefile automatiza la compilación del proyecto so_long.

- NAME := so_long: Define el nombre del ejecutable.
- CC := cc: Define el compilador a utilizar (cc por defecto).
- CFLAGS := -Wall -Wextra -Werror -g: Define las banderas de compilación:
  -Wall: Muestra todas las advertencias.
  -Wextra: Muestra advertencias adicionales.
  -Werror: Trata advertencias como errores.
  -g: Incluye información de depuración.
- SRC_DIR := src, OBJ_DIR := obj: Define las carpetas para código fuente y objetos.
- SRC := $(wildcard $(SRC_DIR)/*.c): Lista todos los archivos fuente .c.
- OBJ := $(patsubst $(SRC_DIR)/%.c, $(OBJ_DIR)/%.o, $(SRC)): Convierte los archivos fuente en objetos.
- LIBFT_DIR, MLX42_DIR: Define las rutas de las bibliotecas.
- INCLUDES: Define las rutas de los headers.
- LIBS: Lista las librerías a enlazar.
- MLX_FLAGS: Banderas necesarias para MLX42.
- all: Compila las bibliotecas y el ejecutable.
- $(OBJ_DIR)/%.o: Regla para compilar cada archivo .c a .o.
- $(NAME): Enlaza los objetos y bibliotecas para crear el ejecutable.
- $(LIBFT), $(MLX42_LIB): Reglas para compilar las bibliotecas si no existen.
- clean, fclean, re: Reglas para limpiar y recompilar el proyecto.
- .PHONY: Declara objetivos que no son archivos.

2. main.c - Flujo Principal del Programa

Función main():
- Inicializa la estructura del juego (init_game_struct).
- Valida los argumentos (check_args).
- Lee el mapa (read_map).
- Valida el tamaño y los elementos del mapa (validate_size_lines_map, validate_elements_map).
- Inicializa el juego (init_game).

Cada una de estas funciones realiza tareas clave para preparar el juego antes de empezar el bucle de eventos.

3. Funciones Clave y su Análisis Detallado

init_game_struct(t_game *game):
Inicializa todos los campos de la estructura t_game a valores por defecto:
- Punteros a NULL, contadores a 0 (jugador, monedas, imágenes).

check_args(int ac, char **av):
- Verifica que se reciba exactamente un argumento.
- Comprueba que la extensión del archivo sea '.ber'.

Proyecto So_Long - Análisis Detallado del Código

read_map(char *av):
- Abre el archivo de mapa.
- Lee las líneas usando get_next_line y ft_strtrim.
- Convierte la lista de líneas en un array bidimensional.
- Valida que el mapa no esté vacío.

validate_size_lines_map(char **map):
- Asegura que todas las filas del mapa tengan la misma longitud.
- Limita el tamaño del mapa a 50 filas y 49 columnas como máximo.

validate_elements_map(char **map):
- Verifica que haya al menos un jugador, una salida y un coleccionable.
- Comprueba que el mapa esté rodeado por muros ('1').
- Valida que todos los coleccionables y la salida sean accesibles (verify_path).

init_game(char **map, t_game *game):
- Inicializa la ventana con MLX42 (init_window).
- Carga los recursos gráficos (load_rscs).
- Renderiza el mapa (render_map).
- Localiza la posición del jugador (search_player).
- Inicia el bucle de eventos (mlx_loop).

move_player, my_keyhook:
- Controlan el movimiento del jugador y verifican si se recolectan objetos o se alcanza la salida.
- Actualizan el conteo de movimientos y la posición del jugador.

4. Conclusión

Este análisis ofrece una comprensión profunda de cómo se estructura y ejecuta el proyecto so_long.
Desde la compilación automatizada hasta la lógica del juego, cada componente interactúa para crear un juego funcional.
Las validaciones aseguran que el mapa sea jugable, y el uso de MLX42 permite renderizar gráficos y manejar eventos.