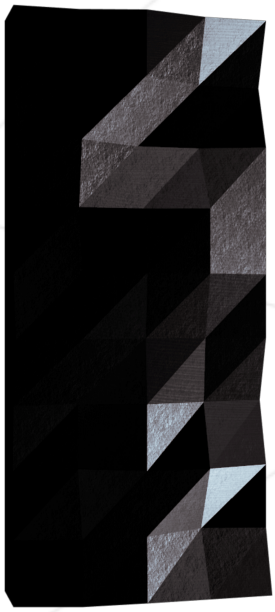

SoLong

And thanks for all the fish!

Resumen: Este proyecto es un pequeño juego en 2D. Está diseñado para hacerte trabajar con texturas y sprites. Por supuesto, elementos básicos de jugabilidad también.

Versión: 1

Índice general

Capítulo I

Avance

Ser desarrollador es genial si quieres crear tu propio juego.
Pero un buen juego necesita recursos gráficos buenos.
Para juegos 2D, debes buscar tiles, tilesets, sprites y sprite sheets.
Por suerte para ti, algunos artistas con talento están dispuestos a compartir su trabajo en plataformas como:
itch.io

Hagas lo que hagas, respeta el trabajo de otros.

Capítulo II

Objetivos

Los objetivos de este proyecto son similares a los de este primer año: rigor, uso de `C`, uso de algoritmos básicos, búsqueda de información, etc.

Como proyecto de diseño gráfico, `so_long` te permitirá mejorar tus habilidades en estas áreas: ventanas, colores, eventos, texturas, etc.

Capítulo III

Instrucciones generales

- Tu proyecto debe estar escrito siguiendo la Norma. Si tienes archivos o funciones adicionales, estas están incluidas en la verificación de la Norma y tendrás un `0` si hay algún error de norma dentro.
- Tus funciones no deben terminar de forma inesperada (segfault, bus error, double free, etc) ni teener comportamientos indefinidos. Si esto pasa tu proyecto será considerado no funcional y recibirás un `0` durante la evaluación.
- Toda la memoria alocada en heap deberá liberarse adecuadamente cuando sea necesario. No se permitirán leaks de memoria.
- Si el subject lo requiere, deberás entregar un `Makefile` que compilará tus archivos fuente al output requerido con las flags `-Wall`, `-Werror` y `-Wextra`, por supuesto tu `Makefile` no debe hacer relink.
- Tu `Makefile` debe contener al menos las normas `$(NAME)`, `all`, `clean`, `fclean` y `re`.
- Para entregar los bonus de tu proyecto, deberás incluir una regla `bonus` en tu `Makefile`, en la que añadirás todos los headers, librerías o funciones que estén prohibidas en la parte principal del proyecto. Los bonus deben estar en archivos distintos `_bonus.{c/h}`. La parte obligatoria y los bonus se evalúan por separado.
- Si tu proyecto permite el uso de la `libft`, deberás copiar su fuente y sus `Makefile` asociados en un directorio `libft` con su correspondiente `Makefile`. El `Makefile` de tu proyecto debe compilar primero la librería utilizando su `Makefile`, y después compilar el proyecto.
- Te recomendamos crear programas de prueba para tu proyecto, aunque este trabajo **no será entregado ni evaluado**. Te dará la oportunidad de verificar que tu programa funciona correctamente durante tu evaluación y la de otros compañeros. Y sí, tienes permitido utilizar estas pruebas durante tu evaluación o la de otros compañeros.
- Entrega tu trabajo a tu repositorio `Git` asignado. Solo el trabajo de tu repositorio `Git` será evaluado. Si Deepthought evalúa tu trabajo, lo hará después de tus compañeros. Si se encuentra un error durante la evaluación de Deepthought, la evalaución terminará.

Capítulo IV

Parte obligatoria -- muy larga

Nombre de programa	`so_long`
Archivos a entregar	`Todos tus archivos`
Makefile	`all, clean, fclean, re, bonus`
Argumentos	`un mapa en formato *.ber`
Funciones autorizadas	• `open, close, read, write, printf, malloc, free, perror, strerror, exit` • `Todas las funciones de la miniLibX`
Se permite usar libft	`Sí`
Descripción	`Debes crear un pequeño juego 2D donde un delfín escape del planeta Tierra después de comer pescado...O cualquier héroe que recolecte objetos de valor antes de abandonar el sitio.`

Los principios son los siguientes:

- Debes usar la `miniLibX`. Ya sea la versión disponible en el sistema operativo, o su fuente. Si eliges trabajar con la fuente, deberás compilar siguiendo las mismas normas que con tu `libft`, descritas en la parte de `Instrucciones generales`.
- La gestión de tu ventana debe ser limpia: cambiar de ventana, minimizar, etc.
- Algunos ejemplos se dan con una temática de delfín, pero puedes usar lo que quieras.

- El mapa estará construido de 3 componentes: paredes, coleccionables y espacio abierto.
- El objetivo del jugador es recolectar todos los coleccionables presentes en el mapa antes de escapar con movimientos mínimos.
- Tras cada movimiento, el número actual de movimientos debe mostrarse en un terminal.
- El jugador debe poder: subir, bajar, ir a la izquierda o ir a la derecha.
- Utilizarás una perspectiva 2D (vista de pájaro o lateral).
- El juego no necesita ser en tiempo real.
- El jugador no puede entrar dentro de las paredes.
- El programa mostrará la imagen en una ventana y respetará las siguientes normas:
  - Las teclas W, A, S y D se utilizarán para mover al personaje principal.
  - Pulsar la tecla `ESC` debe cerrar la ventana y cerrar el programa limpiamente.
  - Hacer clic en la cruz roja de la ventana debe cerrar la ventana y terminar el programa limpiamente.
  - El uso de `images` de la `miniLibX` se recomienda encarecidamente.
- Tu programa debe aceptar como primer argumento un archivo con la descripción del mapa de extensión `.ber`.
  - El mapa estará compuesto de solo 5 caracteres: **0** para un espacio vacío, **1** para un muro, **C** para un coleccionable, **E** para salir del mapa y **P** para la posición inicial del jugador.
    Este es un ejemplo simple de un mapa válido:

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

  - El mapa deberá estar cerrado/rodeado de muros, en caso contrario el programa deberá devolver un error.
  - El mapa debe tener al menos una salida, un coleccionable y una posición inicial.
  - No necesitas comprobar si hay un camino válido en el mapa.
  - El mapa debe ser rectangular.
  - Debes poder procesar cualquier tipo de mapa, siempre y cuando respete las anteriores normas.

- Otro ejemplo minimalista de un mapa **.ber**:

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

- En caso de fallos de configuración de cualquier tipo encontrados en el archivo, el programa debe terminar correctamente y devolver “Error\n” seguido de un mensaje explícito de tu elección.

Capítulo V

Parte extra

Los bonus solo serán evaluados si la parte obligatoria está PERFECTA. Por PERFECTA queremos decir evidentemente que debe estar completa, sin fallos, aunque sea en casos retorcidos o de mal uso, etc. Significa que tu parte obligatoria tiene TODOS los puntos. De otro modo, los bonus serán completamente IGNORADOS.

Se permite el uso de otras funciones para completar la parte extra, siempre y cuando su uso se justifique durante la evaluación. Sé inteligente.

Lista de bonus:

- Patrullas de enemigos que hacen perder al jugador en caso de contacto.
- Algunas animaciones de sprites.
- Un contador de movimiento directamente mostrado en pantalla en lugar de en el terminal.

Capítulo VI

Ejemplos

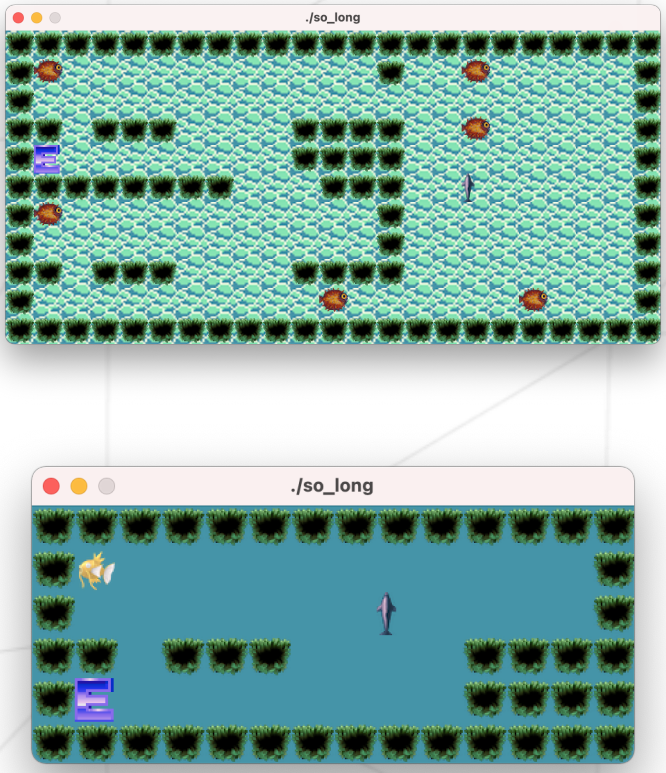

Figura VI.1: Algunos ejemplos de `so_long` con algo de (casi digno de bonus) mal gusto en diseño gráfico.