Algoritmos y Programación I

Curso Camejo

.UBAfiuba
FACULTAD DE INGENIERÍA

TP Segunda Parte

La Sombra de Mac

Fecha Presentación	28/10/2021
Fecha Entrega	25/11/2021

1. Introducción

Mac es un niño de ocho años con gran inteligencia y creatividad. Él es una persona sensata y moral, y puede ser a través de Bloo, su amigo imaginario, que hace y dice todas las cosas que quiere pero no puede. Por lo tanto, Bloo existe como un desafío a la moralidad de Mac. Pasa una gran porción de sus días involucrándose en sus travesuras dentro y fuera de Foster's. La **Mansión Foster** para amigos imaginarios es un inmenso orfanato con estilo victoriano al que los amigos imaginarios se van a vivir cuando ya no pueden pertenecerle a sus creadores. Ahí se muda Bloo, y Mac lo visita todos los días para asegurarse de que Bloo no sea adoptado.

Bloo es el amigo imaginario de Mac, a menudo egoísta, rebelde, inquieto, arrogante o en busca de atención. No obstante, puede cambiar su actitud en lealtad a su creador. Su cuerpo humanoide completamente azul es comparable a la de un guante de cocina o un fantasma del videojuego Pac-Man. A menudo, tiene una tendencia a aceptar ideas escandalosas como un hecho para explicar sucesos aparentemente mundanos, y somete a otros a su voluntad, yendo tan lejos como para hacerlos ir en contra de sus estándares morales. Bloo también muestra pasión por la pelota de pádel, aunque nunca logra hacer que la pelota golpee la paleta, a lo que insiste en que todas sus paletas están rotas.

2. Objetivo

El presente trabajo práctico tiene como objetivo evaluar a los alumnos en aspectos fundamentales de la programación. Estos aspectos son:

- Validación de datos ingresados por el usuario.
- Diseño y desarrollo de funcionalidades de una biblioteca con un contrato preestablecido.
- El correcto uso de estructuras de control.
- Tipos de dato simples y estructurados.
- Buenas prácticas de programación.
- Modularización.

3. Enunciado

Como desarrolladores de este juego, debemos ayudar a Mac y a Bloo a volver de la mansión luego de haber pasado una tarde explorando la ciudad.

Deberás guiarlos en su camino ingresando una dirección en la que caminar, para poder encontrar la llave que abre la puerta de la mansión. Pero cuidado! Bloo, para hacer la vuelta de forma divertida, va a caminar en espejo con respecto a Mac. Es decir, si Mac se dirige en algun sentido horizontal (izquierda o derecha), Bloo irá en el camino contrario.

Para comenzar, Mac **(M)** tendrá 3 vidas disponibles, sus puntos iniciarán en 0 y su posición será aleatoria. En cuanto a Bloo **(B)**, su sombra, empezará estando vivo en la misma fila que Mac, y la columna se calculará como: columnas_totales - columna_mac - 1

Como el camino es largo, y no todo el camino es igual, deberán pasar por 3 niveles distintos, donde cada nivel estará delimitado por los bordes del terreno, y donde habrá paredes, obstáculos y herramientas, que podrán ayudarlos, o no, a volver. En caso de que Bloo o Mac, se choque con algun borde del terreno, o con alguna pared, el mismo no deberá moverse, pero el otro si.

Podrás ayudarlos a volver a la mansión?

3.1. Obstáculos

Los obstáculos son elementos que estarán en cualquier posición del mapa e intentarán matar o distraer a los personajes.

- **Velas:** Dejarán sin vida a la sombra, dejándola inmóvil, por lo que el personaje deberá ir a apagar la vela, y luego revivir a Bloo. El rango de su efecto será de una matriz de 3x3, ubicandose la vela en el centro. Descontarán 50 puntos al personaje, cada vez que este la reviva.
- **Pozos:** Quitarán una vida al personaje en caso de que éste pise uno. Al morir, ambos personajes vuelven a iniciar de 0, es decir, Mac en una coordenada random, y Bloo en una posición dependiente del personaje.

- **Interruptor:** Si Mac pisa un interruptor, lo activará e intercambiará la forma en que se mueve su sombra. Si interruptor_apretado es false, Bloo caminará en sentido espejo. En caso contrario, si es true, caminará en el mismo sentido.
- **Portales:** Si Bloo pisa un portal, las posiciones de los personajes serán intercambiadas, por lo que Bloo quedará en la posición de Mac, y Mac en la de Bloo.

Todos los obstáculos empezarán en posiciones aleatorias. Cabe destacar que no pueden posicionarse distintos obstáculos en la misma posición que otro objeto, o personaje.

El orden de posicionamiento de los obstáculos al inicializar el nivel es indiferente, siempre y cuando se respete lo enunciado en el párrafo anterior.

Cuando Mac apague una vela, se requiere que la vela sea eliminada del vector y no se muestre más en el terreno.

3.2. Herramientas

Las herramientas son elementos que ayudarán a Mac y a Bloo a volver a la mansión.

- **Escaleras:** Se posicionarán sobre las paredes. Tienen un solo uso, por lo que al usarla deberá desaparecer y poner una pared en su lugar. Permitirán a los personajes saltarse una posición.
- **Monedas:** Al agarrar una moneda, le dará puntos al personaje. Los puntos seran un número aleatorio entre 10 y 20, ambos inclusive.
- **Vidas:** El personaje puede elegir intercambiar 200 puntos por una vida al apretar la letra 'V'. Como máximo, el personaje puede contar con 3 vidas al mismo tiempo.
- **Llave:** Sólo puede ser agarrada por Mac. Le permitirá a ambos personajes terminar el nivel, al pasar por la puerta de la mansión.

El hecho de agarrar cualquiera de estos objetos, con excepción de las vidas, está dado por pararse en la posición en la que se encuentra la herramienta.

Las herramientas deben posicionarse aleatoriamente al inicializar el juego, cabe destacar que no pueden posicionarse distintas herramientas en la misma posición o en una posición donde ya existe un obstáculo, o la posición de uno de los dos personajes.

El orden de posicionamiento de las herramientas al inicializar el juego es indiferente, siempre y cuando se respete lo enunciado en el párrafo anterior.

El agarrar una moneda o una llave, o el uso de la escalera, requerirá que se elimine del vector de herramientas.

3.3. Niveles

Habrá 3 niveles a lo largo de todo el juego. Para pasar cada uno, Mac deberá contar con la llave y ambos personajes deberán estar a una distancia manhattan menor a uno, de la puerta. En cada nivel habrá distinta cantidad de herramientas y obstáculos.

3.3.1. Nivel 1

- Cantidad obstáculos:
  - **Velas**: 5
  - **Pozos**: 15
  - **Interruptores**: 1
  - **Portales**: 0
- Cantidad herramientas:
  - **Escaleras**: 10
  - **Monedas**: 10
  - **Llave**: 0

3.3.2. Nivel 2

- Cantidad obstáculos:
  - **Velas**: 10
  - **Pozos**: 20
  - **Interruptores**: 2
  - **Portales**: 2
- Cantidad herramientas:
  - **Escaleras**: 15
  - **Monedas**: 15
  - **Llave**: 1

3.3.3. Nivel 3

- Cantidad obstáculos:
  - **Velas**: 12
  - **Pozos**: 30
  - **Interruptores**: 4
  - **Portales**: 4
- Cantidad herramientas:
  - **Escaleras**: 15
  - **Monedas**: 15
  - **Llave**: 1

4. Especificaciones

Como buenos amigos, vamos a ayudar a Mac y Bloo a cumplir su objetivo de volver a la mansión. Para poder lograr esto, se pedirá implementar algunas funciones y procedimientos.

4.1. Funciones y procedimientos

```c
#ifndef __LA_SOMBRA_DE_MAC__
#define __LA_SOMBRA_DE_MAC__

#include <stdbool.h>

#define MAX_FILAS 20
#define MAX_COLUMNAS 25
#define MAX_ELEMENTOS 500
#define MAX_NIVELES 3
#define MAX_PAREDES 200

typedef struct coordenada {
    int fila;
    int col;
}coordenada_t;

typedef struct personaje {
    coordenada_t posicion;
    int vida;
    int puntos;
    bool tiene_llave;
    bool interruptor_apretado;
}personaje_t;

typedef struct sombra {
    coordenada_t posicion;
    bool esta_viva;
}sombra_t;
```

```c

typedef struct elemento {
    char tipo;
    coordenada_t coordenada;
}elemento_t;

typedef struct nivel {
    int numero_nivel;
    coordenada_t paredes[MAX_PAREDES];
    int tope_paredes;
    elemento_t obstaculos[MAX_ELEMENTOS];
    int tope_obstaculos;
    elemento_t herramientas[MAX_ELEMENTOS];
    int tope_herramientas;
}nivel_t;

typedef struct juego {
    personaje_t personaje;
    sombra_t sombra;
    nivel_t niveles[MAX_NIVELES];
    int nivel_actual;
}juego_t;


/*
 * Procedimiento que recibe el juego e imprime toda su información por pantalla.
 */
void imprimir_terreno(juego_t juego);

/*
 *Inicializará el juego, cargando toda la información inicial, los datos del personaje, y los 3
    niveles.
 */
void inicializar_juego(juego_t* juego);

/*
 * Recibe un juego con todas sus estructuras válidas.
 *
 * El juego se dará por ganado, si terminó todos los niveles. O perdido, si el personaje queda
 * sin vida.
 * Devolverá:
 * -> 0 si el estado es jugando.
 * -> -1 si el estado es perdido.
 * -> 1 si el estado es ganado.
 */
int estado_juego(juego_t juego);

/*
 * El nivel se dará por terminado, si ambos personajes pasaron por la puerta teniendo la
 * llave correspondiente.
 * Devolverá:
 * -> 0 si el estado es jugando.
 * -> 1 si el estado es ganado.
 */
int estado_nivel(juego_t juego);

/*
 * Moverá el personaje, y realizará la acción necesaria en caso de chocar con un elemento
 */
void realizar_jugada(juego_t* juego);

#endif
```

Observación: Queda a critero del alumno/a el hacer o no, más funciones y/o procedimientos para resolver los problemas presentados. No se permite agregar dichas firmas al .h. Algunas funciones y procedimientos, ya no se encuetran más en la_sombra_de_mac.h debido a que ahora pasarán a ser privadas.

4.2. Convenciones

Se deberá utilizar la siguiente convención para los obstáculos y herramientas:

- **Puerta**: D.
- **Escaleras**: E.

- **Llave**: L.
- **Monedas**: C.
- **Pozos**: W.
- **Interruptores**: O.
- **Portales**: P.
- **Velas**: V.

Y para los personajes:

- **Mac**: M.
- **Bloo**: B.

5. Resultado esperado

Se espera que el trabajo creado cumpla con las buenas prácticas de programación y todas las funciones y procedimientos pedidos funcionen acorde a lo solicitado, respetando las pre y post condiciones propuestas.

6. Compilación y Entrega

El trabajo práctico debe ser realizado en un archivo llamado la_sombra_de_mac.c, lo que sería la implementación de la biblioteca la_sombra_de_mac.h. El objetivo es que sea compilado sin errores al correr desde la terminal el comando:

```
gcc *.c utiles.o -o juego -std=c99 -Wall -Wconversion -Werror -lm
```

utiles.o es un archivo compilado realizado por la cátedra, que pondrá a su disposición funciones que pueden ser, justamente, útiles y su funcionamiento se explica en el anexo.

Por último debe ser entregado en la plataforma de corrección de trabajos prácticos **Chanutron2021** (patente pendiente), en la cual deberá tener la etiqueta **¡Exito!** significando que ha pasado las pruebas a las que la cátedra someterá al trabajo.

Para la entrega en **Chanutron2021** (patente pendiente), recuerde que deberá subir un archivo **zip** que contenga únicamente los archivos antes mencionados, sin carpetas internas ni otros archivos. De lo contrario, la entrega no será validada por la plataforma.

IMPORTANTE! Obtener la etiqueta **¡Exito!** en **Chanutron2021** (patente pendiente) no implica necesariamente haber aprobado el trabajo. El trabajo será corregido por un colaborador que verificará que se cumplan las buenas prácticas de programación.

7. Anexos

7.1. Obtención de números aleatorios

Para obtener números aleatorios debe utilizarse la función **rand()**, la cual está disponible en la biblioteca stdlib.h.

Esta función devuelve números pseudo-aleatorios, esto quiere decir que, cuando uno ejecuta nuevamente el programa, los números, aunque aleatorios, son los mismos.

Para resolver este problema debe inicializarse una semilla, cuya función es determinar desde donde empezarán a calcularse los números aleatorios.

Los números arrojados por **rand()** son enteros sin signo, generalmente queremos que estén acotados a un rango (queremos números aleatorios entre tal y tal). Para esto, podemos obtener el resto de la división de **rand()** por el valor máximo del rango que necesitamos.

Aquí dejamos un breve ejemplo de como obtener números aleatorios entre 10 y 29 (inclusives).

```c
#include <stdio.h>
#include <stdlib.h> // Para usar rand
#include <time.h>   // Para obtener una semilla desde el reloj

int main(){
        srand ((unsigned)time(NULL)); // Genera la semilla aleatoria.
        int numero = rand() % 20 + 10; // La amplitud del rango es 20 y el valor mínimo es 10.
        printf("El valor aleatorio es: %i\n", numero);

        return 0;
}
```

7.2. Distancia Manhattan

Para obtener la distancia entre 2 puntos mediante este método, se debe conocer a priori las coordenadas de dichos puntos.

Luego, la distancia entre ellos es la suma de los valores absolutos de las diferencias de las coordenadas. Se ve claramente en los siguientes ejemplos:

- La distancia entre los puntos (0,0) y (1,1) es 2 ya que: $|0 - 1| + |0 - 1| = 1 + 1 = 2$
- La distancia entre los puntos (10,5) y (2,12) es 15 ya que: $|10 - 2| + |5 - 12| = 8 + 7 = 15$
- La distancia entre los puntos (7,8) y (9,8) es 2 ya que: $|7 - 9| + |8 - 8| = 2 + 0 = 2$

7.3. Limpiar la pantalla durante la ejecución de un programa

Muchas veces nos gustaría que nuestro programa pueda verse siempre en la pantalla sin ver texto anterior.

Para ésto, podemos utilizar la llamada al sistema **clear**, de esta manera, limpiaremos todo lo que hay en nuestra terminal hasta el momento y podremos dibujar la información actualizada.

Y se utiliza de la siguiente manera:

```c
#include <stdio.h>
#include <stdlib.h>

int main(){
        printf("Escribimos algo\n");
        printf("que debería\n");
        printf("desaparecer...\n");

        system("clear"); // Limpiamos la pantalla

        printf("Solo deberiamos ver esto...\n");
        return 0;
}
```

Referencias

Links:

https://fostershomeforimaginaryfriends.fandom.com/wiki/Mac
https://fostershomeforimaginaryfriends.fandom.com/wiki/Bloo