Algoritmos y Programación I
Curso Mendez

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Helm Royale

Parte II - La Batalla

Fecha de Presentación: 26/09/2019

Fecha de Vencimiento: 24/10/2019

Un Anillo para gobernarlos a todos.
Un Anillo para encontrarlos, un Anillo para atrerlos a todos y atarlos a las tinieblas
en la Tierra de Mordor donde se extienden las Sombras...

1. Introducción

El Abismo de Helm era un desfiladero que se abría paso entre las Ered Nimrais bajo el Thrihyrne. El Abismo de Helm era el centro defensivo del Folde Oeste de Rohan; éste y El Sagrario eran las principales fortalezas y refugios del reino. La Corriente del Bajo salía del Abismo de Helm, y allí se hallaban las Aglarond. El Muro del Bajo se levantó para cerrar la entrada al Abismo.

Durante la Guerra del Anillo los hombres de Rohan se refugiaron en el Abismo de Helm del asedio del ejército de Saruman el Blanco, formado por enormes fuerzas, más de 10.000 Uruk-Hai, orcos, semiorcos y huargos.

El objetivo de las fuerzas de Isengard era aniquilar totalmente al pueblo de Rohan de un golpe, y dejar a Gondor sin aliados formales ante las legiones de Sauron.

Algunos héroes conocidos de esta batalla son Aragorn, Legolas y Charly, además de Théoden, rey de Rohan y Éomer su sobrino y heredero.

Las huestes de Saruman llegaron al Valle del Abismo de Helm en medio de la noche, y muy rápido escalaron la primera defensa e intentaron derribar la puerta de la fortaleza con un ariete. Pero Aragorn, Éomer y algunos otros rohirrim atacaron, a través de una puerta trasera en el lado de Horburg, dispersando a los orcos que atacaban las puertas.

Los orcos y los dunledinos levantaron entonces cientos de escaleras para escalar el muro de la fortaleza. Aragorn y Éomer tuvieron que mover repetidas veces a los defensores, que estaban agotados, para poder repeler a los orcos, subiendo escaleras, y cruzando el muro de un lado a otro. Sin embargo, algunos orcos pudieron infiltrarse a través de una pequeña alcantarilla, que permitía que una corriente de agua saliese del Abismo de Helm, y mientras los rohirrim estaban repeliendo a los orcos que escalaban el muro, intentaron llegar a ella luego de superar una barrera. Los defensores reaccionaron rápidamente y expulsaron a los orcos, y la alcantarilla fue bloqueada bajo la supervisión de Charly.

Pero los orcos volvieron a entrar en la alcantarilla, y gracias a un dispositivo ideado por Saruman, causaron una gran explosión, creando un enorme agujero en el muro del Abismo.

Los orcos pudieron entrar sin poder ser detenidos. Los defensores se retiraron a las Cuevas Relucientes y al Hornburg. Pronto las fuerzas de Saruman usaron sus armas para ganar la entrada a la fortaleza, que antes había defendido Aragorn.

La batalla no ha terminado aún, es momento de tomar partida, para uno u otro bando...

2. Objetivo

El presente trabajo práctico tiene como objetivo que el alumno:

- Diseñe y desarrolle las funcionalidades de una biblioteca con un contrato preestablecido.
- Se familiarice con y utilice correctamente los tipos de datos estructurados.
- Desarrolle una interfaz gráfica amigable y entendible para el usuario.

Por supuesto, se requiere que el trabajo cumpla con las buenas prácticas de programación profesadas por la cátedra.

Se considerarán críticos la modularización, reutilización de código y la claridad del código.

3. Enunciado

La 2da parte de Helm Royale consiste en desarrollar un programa que permita recrear la épica batalla del abismo de Helm.

El juego, se desarrolla sobre un terreno rectangular, dividido en 2, donde la mitad inferior corresponde al perfil defensivo, en adelante Rohan y la mitad superior al perfil ofensivo, en adelante Isengard.

Rohan posicionará soldados en su mitad, que intentarán destruir a Isengard, y por su parte, Isengard posicionará soldados en su mitad que intentarán destruir a Rohan.

Tanto Rohan cómo Isengard cuentan con 2 tipos de soldados cada uno. Rohan tiene Elfos y Hombres, mientras que Isengard tendrá Uruk-hai y Orcos.

Cada una de las partes tendrá una barra de energía que no superará los 10 puntos, con esa energía puede posicionar soldados en el campo.

La partida termina cuando 5 hombres llegan a Isengard (en ese caso gana Rohan) o cuando 5 Orcos llegan a Rohan (en ese caso gana Isengard).

3.1. Terreno

El terreno o campo de juego debe verse como una matriz, la mitad inferior corresponde a Rohan y la superior a Isengard.

El tamaño será de 10x10.

3.2. Personajes

Para cada personaje creado deberá obtenerse un número aleatorio que, multiplicado por la intensidad, serán claves en su ataque y vida.

A ese número le llamaremos plus y se calcula cómo:

plus = intensidad * (numero aleatorio entre 0 y 5).

3.2.1. Elfos y Uruk-hai

Debido a su gran habilidad con el arco y la flecha, estos personajes tomarán posiciones fijas en el terreno, defendiendo su mitad.

Rohan comenzará con 3 elfos posicionados e Isengard con 3 Uruk-hai. Estas posiciones serán aleatorias dentro de cada mitad correspondiente.

No puede haber más de un Elfo en la misma posición y no puede haber más de un Uruk-hai en la misma posición.

Las armas que utilizan estos personajes permiten atacar a distancia, por lo que pueden atacar a rivales que se encuentren hasta una distancia de 3 posiciones según la Distancia Manhattan (ver anexo).

Adicionalmente, por cada turno, estos soldados atacan a todos los que tienen en su rango, ya que son muy rápidos con sus armas.

Cada Elfo y cada Uruk-hai cuestan 8 puntos de energía.

Durante la partida el jugador puede elegir en qué coordenadas posicionar nuevos elfos, siempre y cuando sea dentro de la mitad que le corresponde y no haya otro soldado del mismo tipo en dicha posición.

Adicionalmente, los Elfos no pueden posicionarse, ni en la inicialización, ni durante la partida, en la última fila, lo mismo para los Uruk-hai pero en la primer fila.

Ataque: 10 pts. + plus.

Vida: 200 pts. - plus.

3.2.2. Hombres y Orcos

Los Hombres y Orcos no tendrán la certeza de los personajes anteriores, pero pueden moverse por el campo con gran destreza y blandir la espada con una letalidad increíble. Estos personajes aparecerán en el terreno cuando cada jugador así lo disponga, si es que cuenta con la energía necesaria para crearlos.

Su objetivo es simple, los Hombres 'subirán' y los Orcos 'bajarán' por la matriz en línea recta.

Cuando se crucen con algún enemigo a si alrededor (posiciones adyacentes) no podrán evitar combatir hasta que uno de ellos desista.

Tener en cuenta que, revolear una espada no es algo sencillo, por lo que estos personajes solo pueden atacar a un rival por turno.

Los Hombres que choquen con Elfos, los pasarán por encima y los Orcos que choquen con Uruk-hai harán lo mismo.

Rohan podrá posicionar sus Hombres en la última fila de la matriz, en la columna que decida. Isengard, por su parte, posicionará a los Orcos en la primer fila de la matriz, también en la columna que decida.

Cada uno de estos personajes se mueven un casillero por turno y solo lo harán si no existe un adversario alrededor.

Cada Hombre y cada Orco cuestan 3 puntos de energía.

Ataque: 50 pts. + plus.

Vida: 100 pts. - plus.

3.3. Puntos de Energía

Tanto Rohan cómo Isengard comenzarán con 5 puntos de energía, que pueden utilizarlos o no según deseen para posicionar personajes.

Cada turno que concluye les da un punto de energía a cada uno.

El máximo de puntos de energía que puede acumularse es 10, los puntos que se obtengan cuando se supere este valor, se perderán.

3.4. Turnos

Cada turno consiste en los siguientes eventos:

- Rohan decide si posiciona o no un personaje.
- Isengard decide si posiciona o no un personaje.
- Los elfos de Rohan atacan.
- Los Uruk-hai de Isengard atacan.
- Los Hombres de Rohan se mueven o atacan.
- Los Orcos de Isengard se mueven o atacan.

Luego de cada turno se le otorgará 1 punto de energía a cada jugador.

Los turnos se repetirán hasta que la partida finalice.

3.5. Determinación de Bandos

Se deberá utilizar la funcionalidad desarrollada en la Parte I para determinar de qué lado y con qué intensidad jugará el primer jugador.

El 2do, automáticamente será del bando contrario y con la intensidad faltante para llegar a 10.

Por ejemplo:

- Si el primer jugador es ofensivo con intensidad 4, el 2do jugador será defensivo con intensidad 6.
- Si el primer jugador es defensivo con intensidad 10, el 2do jugador será ofensivo con intensidad 0.

3.6. Jugadores

La partida puede jugarse entre 2 jugadores o 1 contra la máquina.

La determinación de los bandos se hará cómo se explica en la sección anterior. Siempre comenzará el jugador que le haya tocado Rohan.

En caso de jugar contra la máquina, ésta tomará siempre la decisión de poner un personaje cuando la energía se lo permita, es decir que si juega con Rohan, cada vez que tenga 3 puntos de poder, posicionará un Hombre en el terreno, y si es Isengard, posicionará un Orco.

La columna en la que empezará el personaje deberá determinarse aleatoriamente.

El juego, debe consultar al usuario si la partida será de a 2 o si será contra la máquina.

4. Especificaciones

4.1. Estructuras de Datos

Pueden crearse todas las estructuras que desee y se deberá mantener actualizada la siguiente estructura a lo largo de todo el trabajo.

```
typedef struct personaje {
        char codigo;
        int vida;
        int ataque;
        int fila;
        int columna;
} personaje_t;

typedef struct juego {
        char terreno[MAX_TERRENO_FIL][MAX_TERRENO_COL];
        personaje_t rohan[MAX_PERSONAJES];
        int cantidad_rohan;
        int llegadas_rohan;
        int plus_rohan;
        personaje_t isengard[MAX_PERSONAJES];
        int cantidad_isengard;
        int llegadas_isengard;
        int plus_isengard;
} juego_t;
```

4.2. Convenciones

Se deberá utilizar, para los distintos personajes, la siguiente convención:

- Elfos: E.
- Hombres: H.
- Uruk-hai: U.
- Orcos: O.

4.3. Customización

Si bien en las secciones anteriores hemos especificado ciertos valores, es imprescindible que algunos de ellos puedan ser cambiados fácilmente y se vean reflejados en el juego sin demasiado esfuerzo, estos son:

- Tamaño del terreno.
- Costo de crear cada personaje.

- Cantidad de Elfos y Uruk-hai al comienzo de la partida.
- Cantidad de personajes que deben llegar al lado contrario para ganar la partida.

4.4. Bibliotecas

4.4.1. perfil.h

Se debe crear una biblioteca con el trabajo realizado en la Parte 1, ésta biblioteca solo tendrá un procedimiento con la siguiente firma:

```
void perfil(char* tipo, int* intensidad);
```

4.4.2. batalla.h

Se debe crear la biblioteca batalla donde se implementarán una serie de funciones y procedimientos propuestos por la cátedra y donde se podrán agregar todas las funciones y procedimientos que el alumno considere necesarios.

Las funciones y procedimientos imprescindibles son:

```
/*
 * Inicializará todos los valores del juego, dejándolo en un estado
 * inicial válido.
 */
void inicializar_juego(juego_t* juego);

/*
 * Recibirá un personaje, con todos sus campos correctamente cargados y
 * lo dará de alta en el juego, sumándolo al vector correspondiente,
 * posicionándolo también en la matriz.
 */
void posicionar_personaje(juego_t* juego, personaje_t personaje);

/*
 * Realizará la jugada del personaje del bando recibido que
 * se encuentra en la posición posicion_personaje.
 * Se moverá o atacará dependiento lo que corresponda.
 * Actualizará el juego según los efectos del movimiento del
 * personaje, matar rivales, actualizar la matriz, restar vida, etc.
 */
void jugar(juego_t* juego, char bando, int posicion_personaje);
```

5. Resultado esperado

Se espera que se creen las funciones y procedimientos para poder jugar el juego con fluidez.

Muchas de las funcionalidades quedan a criterio del alumno, solo se pide que se respeten las estructuras y funciones pedidas.

El trabajo creado debe:

- Interactuar con el usuario.
- Mostrarle al/los jugador/es la información del juego a cada momento.
- Informarle al/los jugador/es correctamente cualquier dato que haya/n sido ingresado incorrectamente.
- Informarle al/los jugador/es quien ganó.
- Cumplir con las buenas prácticas de programación.
- Mantener las estructuras propuestas actualizadas a cada momento.

6. Compilación y Entrega

El trabajo práctico debe ser realizado en un archivo llamado **juego.c**, y la biblioteca de funciones para jugarlo **batalla.c** y **batalla.h**, y debe poder ser compilado sin errores con el comando:

```
gcc juego.c batalla.c perfil.c -o juego -std=c99 -Wall -Wconversion -Werror -lm
```

perfil.c y **perfil.h** corresponden a la biblioteca creada con la parte 1 del trabajo para obtención del perfil del jugador

Por último debe ser entregado en la plataforma de corrección de trabajos prácticos Kwyjibo en la cual deberá aparecer con la etiqueta successful.

Para la entrega en Kwyjibo, recuerde que deberá subir un archivo zip conteniendo únicamente los archivos antes mencionados, sin carpetas internas ni otros archivos. De lo contrario, la entrega no será validada por la plataforma.

7. Anexo

7.1. 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: $|7 - 9| + |8 - 8| = 2 + 0 = 2$

7.2. 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 mismo.

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 acotdos a un rango (queremos números aleatorios entre tal y tal). Para ésto, 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 30.

```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));
        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;
}
```

7.3. 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

https://bibliotecadelatierramedia.fandom.com/es/wiki/Batalla_del_Abismo_de_Helm
https://tolkiendili.com/wiki/Batalla_del_Abismo_de_Helm