#include "include/menu.h"

/\*\*

\* Función principal para el menú del juego. Esta función se encarga de llamar al resto

\* en bucle.

\*

\* @param malla

\* Estructura con la información del juego.

\*/

void menu (Malla malla)

{

int selecc = 0;

bool fin = false;

while (!fin)

{

mostrar\_malla (malla);

/\* Imprime el menú y permite elegir opciones \*/

imprimir (DETALLE\_LOG, MSG\_MENU);

/\* Pide la opción seleccionada (entre el 0 y el 6) \*/

selecc = pedir\_opcion (0, 6);

/\*

Opciones disponibles:

0.- Salir

1.- Mover diamante

2.- Bomba

3.- Guardar partida

4.- Cargar partida

5.- Cambiar nivel

6.- Jugada automática

\*/

switch (selecc)

{

case 0:

fin = true;

break;

case 1:

mover (malla);

break;

case 2:

bomba (malla);

break;

case 3:

guardar\_partida (malla);

break;

case 4:

cargar\_partida (&malla);

break;

case 5:

cambiar\_nivel (&malla);

break;

case 6:

realizar\_jugada (&malla);

break;

default:

imprimir (DETALLE\_LOG, "Opción no reconocida.\n");

}

if (!fin)

{

eliminar\_coincidencias (&malla);

imprimir (DETALLE\_DEBUG, "Estado de la matriz tras eliminar"

" coincidencias: \n");

if (ver\_nv\_detalle () >= DETALLE\_DEBUG)

{

mostrar\_malla (malla);

}

llenar\_vacios (&malla);

}

}

}

/\*\*

\* Pide los datos necesarios para realizar un movimiento.

\*

\* @param malla

\* Estructura con la matriz en la que se van a mover los elementos.

\*/

void mover (Malla malla)

{

int posY = 0,

posX = 0,

mov = 0;

/\* Pide los datos \*/

imprimir (DETALLE\_LOG, "Posición del elemento a mover:\n");

/\* Fila \*/

imprimir (DETALLE\_LOG,

"\tFila (entre 0 y %i): ",

(malla.dimens.filas - 1));

posY = pedir\_opcion (0, (malla.dimens.filas - 1));

/\* Columna \*/

imprimir (DETALLE\_LOG,

"\tColumna (entre 0 y %i): ",

(malla.dimens.columnas - 1));

posX = pedir\_opcion (0, (malla.dimens.columnas - 1));

/\* Movimiento \*/

imprimir (DETALLE\_LOG, "Movimiento. Opciones disponibles:\n"

"\t%i -> derecha\n"

"\t%i -> abajo\n"

"\t%i -> izquierda\n"

"\t%i -> arriba\n"

"Introduzca el movimiento seleccionado: ",

MOV\_DER,

MOV\_ABAJO,

MOV\_IZQ,

MOV\_ARRIBA);

mov = pedir\_opcion (MOV\_DER, MOV\_ARRIBA);

/\* Realiza el movimiento y comprueba si hay elementos alineados \*/

mover\_diamante (posY, posX, mov, malla);

}

/\*\*

\* Pide los datos necesarios para usar una bomba.

\*

\* @param malla

\* Estructura con la matriz en la que se van a mover los elementos.

\*/

void bomba (Malla malla)

{

int bomba = 1,

fila = 0,

columna = 0;

imprimir (DETALLE\_LOG, "Bombas disponibles:\n"

"\t1 -> Eliminar fila\n"

"\t2 -> Eliminar columna\n"

"\t3 -> Girar en grupos de 3x3\n"

"Introduzca la bomba seleccionada: ");

bomba = pedir\_opcion (1, 3);

switch (bomba)

{

case 1:

imprimir (DETALLE\_LOG,

"Fila (entre 0 y %i): ",

(malla.dimens.filas - 1));

fila = pedir\_opcion (0, (malla.dimens.filas - 1));

/\* Llama a la función que utiliza CUDA \*/

bomba\_fila (fila, &malla);

break;

case 2:

imprimir (DETALLE\_LOG,

"Columna (entre 0 y %i): ",

(malla.dimens.columnas - 1));

columna = pedir\_opcion (0, (malla.dimens.columnas - 1));

/\* Llama a la función que utiliza CUDA \*/

bomba\_columna (columna, &malla);

break;

case 3:

/\* Llama a la función que utiliza CUDA \*/

bomba\_giro (&malla);

break;

}

}

/\*\*

\* Pide una opción por teclado y muestra un mensaje de error hasta que se ha introducido

\* un número válido en el rango especificado.

\*

\* @param min

\* Rango inferior del grupo de números admitidos (inclusivo).

\*

\* @param max

\* Rango superior del grupo de números admitidos (inclusivo).

\*

\*

\* @return

\* El valor introducido por teclado.

\*/

int pedir\_opcion (int min, int max)

{

int selecc = -1,

ret\_val = -1;

char entrada [50];

do

{

/\* Obtiene la entrada por teclado y busca un número. Si no lo encuentra,

descarta la entrada hasta que se introduzca un número. \*/

fgets (entrada, sizeof entrada, stdin);

ret\_val = sscanf (entrada, "%d", &selecc);

if (ret\_val != 1)

{

imprimir (DETALLE\_LOG,

"Entrada incorrecta. Introduzca el número"

" de la opción seleccionada: ");

}

else if ((selecc < min)

|| (selecc > max))

{

ret\_val = -1;

imprimir (DETALLE\_LOG,

"Sólo se admiten números entre %i y %i.\n"

"Introduzca de nuevo un valor: ",

min, max);

}

} while (ret\_val != 1);

return selecc;

}

/\*\*

\* Muestra las opciones para guardar la partida actual.

\*

\* @param malla

\* Estructura de tipo Malla (definida en 'common.h') con la información de

\* la partida.

\*/

void guardar\_partida (Malla malla)

{

char fichero [100];

/\* Pide el nombre del fichero \*/

imprimir (DETALLE\_LOG, "Introduzca el nombre del archivo en el que"

" desea guardar los datos: ");

fgets (fichero, sizeof fichero, stdin);

/\* Sustituye el salto de línea final \*/

fichero [strcspn (fichero, "\r\n")] = 0;

/\* Guarda la partida \*/

if (guardar (malla, fichero) == SUCCESS)

{

imprimir (DETALLE\_LOG,

"Partida guardada en el archivo '%s'.\n",

fichero);

}

else

{

imprimir (DETALLE\_LOG,

"Error al guardar la partida en '%s'.\n",

fichero);

}

}

/\*\*

\* Muestra las opciones para cargar un fichero con una partida.

\*

\* @param antigua

\* Malla con la información de la partida actual. Si el fichero se carga

\* correctamente, se sustituye por ella. Si no, se deja como estaba.

\*/

void cargar\_partida (Malla \*antigua)

{

Malla nueva;

char fichero [100];

/\* Pide el nombre del fichero \*/

imprimir (DETALLE\_LOG, "Introduzca el nombre del archivo a cargar: ");

fgets (fichero, sizeof fichero, stdin);

/\* Sustituye el salto de línea final \*/

fichero [strcspn (fichero, "\r\n")] = 0;

if (cargar (&nueva, fichero) == SUCCESS)

{

imprimir (DETALLE\_LOG,

"Partida cargada desde '%s' correctamente.\n",

fichero);

antigua->matriz = nueva.matriz;

antigua->dimens = nueva.dimens;

antigua->nivel = nueva.nivel;

cambiar\_params (nueva);

}

else

{

imprimir (DETALLE\_LOG,

"Error al cargar la partida del archivo '%s'.\n",

fichero);

}

}

/\*\*

\* Cambia el nivel del juego.

\*

\* @param malla

\* Estructura (definida en 'common.h') en la que se almacena la información

\* del juego y que va a ser cambiada.

\*/

void cambiar\_nivel (Malla \*malla)

{

int nuevo\_nivel = malla->nivel;

/\* Pide un número para cambiar el nivel \*/

imprimir (DETALLE\_LOG, "Introduzca el nuevo nivel (entre 1 y %i): ", MAX\_NV);

nuevo\_nivel = pedir\_opcion (1, MAX\_NV);

imprimir (DETALLE\_LOG, "Nuevo nivel: %i.\n", nuevo\_nivel);

malla->nivel = nuevo\_nivel;

cambiar\_params (\*malla);

}