PRÁCTICA 2

USO DE BIBLIOTECAS DE PROGRAMACIÓN DE INTERFACES DE USUARIO DE MODO TEXTO

RUBÉN HIDALGO TROYANO
MARÍA MEGÍAS MOYANO

ÍNDICE

1. Instalación de la librería ncurses

Para instalar la librería en Linux hemos usado el sistema de gestión de paquetes correspondiente, con la orden proporcionada para linux.

```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~$ sudo apt-get install libncurses5-dev libncursesw5-dev
[sudo] contraseña para maria:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  dctrl-tools dkms libgsoap-2.8.117 liblzf1 libqt5help5 libqt5opengl5
  libqt5sql5 libqt5sql5-sqlite libqt5x11extras5 libqt5xml5 libwpe-1.0-1
  libwpebackend-fdo-1.0-1
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes NUEVOS:
  libncurses5-dev libncursesw5-dev
0 actualizados, 2 nuevos se instalarán, 0 para eliminar y 152 no actualizados.
Se necesita descargar 1.580 B de archivos.
Se utilizarán 12,3 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libncurses5-dev amd64 6.3-2ubuntu0.1 [790 B]
Des:2 http://es.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libncursesw5-dev amd64 6.3-2ubuntu0.1 [790 B]
Descargados 1.580 B en 0s (7.464 B/s)
Seleccionando el paquete libncurses5-dev:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 316174 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../libncurses5-dev_6.3-2ubuntu0.1_amd64.deb ...
Desempaquetando libncurses5-dev:amd64 (6.3-2ubuntu0.1) ...
Seleccionando el paquete libncursesw5-dev:amd64 previamente no seleccionado.
Preparando para desempaquetar .../libncursesw5-dev_6.3-2ubuntu0.1_amd64.deb ...
Desempaquetando libncursesw5-dev:amd64 (6.3-2ubuntu0.1) ...
Configurando libncursesw5-dev:amd64 (6.3-2ubuntu0.1) ...
Configurando libncurses5-dev:amd64 (6.3-2ubuntu0.1) ...
```

2. Creación de los programas

Cada uno de los ejemplos ofrecidos se han compilado y ejecutado de la siguiente manera:

hello.c

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc hello.c -o hello -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./hello
```

ventana.c

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc ventana.c -o ventana -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./ventana
```

pelotita.c

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc pelotita.c -o pelotita -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./pelotita
```

aventura.c

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc aventura.c -o aventura -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./aventura
```

colores.c

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc colores.c -o colores -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./colores
```

pruncurses.c

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc pruncurses.c -o pruncurses -lncurses
pruncurses.c:10:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   10 | main() {
      | ^~~~
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./pruncurses
```

rebota.c

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc rebota.c -o rebota -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./rebota
```

rebota2.c

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc rebota2.c -o rebota2 -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./rebota2
```

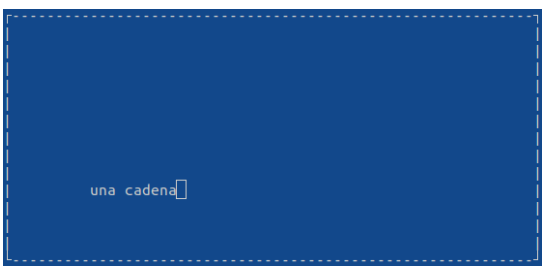
3. Comprobación de su funcionamiento

hello.c

Este programa es un ejemplo básico del uso de la biblioteca ncurses en C, que permite crear interfaces de texto interactivas en la terminal. Al ejecutarse, primero se inicializa la pantalla con initscr(), lo cual prepara el entorno de ncurses. Luego, con printw("Hello World!"), se escribe el texto "Hello World!" en la pantalla. La función refresh() actualiza la pantalla para que el texto realmente se muestre al usuario. getch() detiene la ejecución del programa y espera a que el usuario presione una tecla, permitiendo ver el mensaje antes de salir. Finalmente, endwin() restaura el estado normal de la terminal antes de finalizar el programa. Es un programa simple que demuestra cómo ncurses reemplaza a printf para trabajar directamente con la terminal en modo pantalla completa.

ventana.c

Este programa utiliza la biblioteca ncurses para crear una ventana de texto con colores y bordes dentro de la terminal. Al comenzar, inicializa la pantalla con initscr() y verifica si el terminal soporta colores con has_colors(). Si no los soporta, finaliza el programa con un mensaje. Luego, con start_color() e init_pair(), define combinaciones de colores (texto y fondo). A continuación, obtiene el tamaño actual de la pantalla con getmaxyx() y crea una nueva ventana que ocupa toda el área usando newwin(). A esta ventana se le asigna un fondo azul con texto blanco (COLOR_PAIR(3)) usando wbkgd(), y se dibuja un borde con box(). Después, se imprime el texto "una cadena" en la posición (10, 10) dentro de la ventana con mvwprintw() y se actualiza la ventana con wrefresh(). Finalmente, espera una tecla con getch() y sale limpiamente con endwin(). El objetivo del programa es mostrar cómo usar ventanas, colores y bordes personalizados en ncurses.

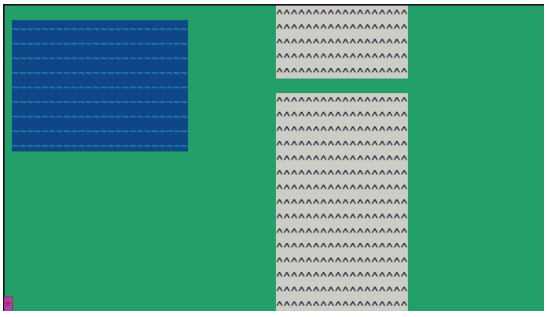


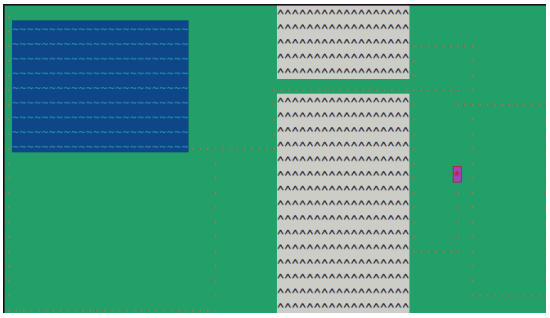
pelotita.c

Este programa crea una animación simple usando ncurses, donde una pelotita (o) se mueve horizontalmente por la pantalla y rebota en los bordes izquierdo y derecho. Al comenzar, se inicializa la pantalla con initscr(), se desactiva la impresión automática de caracteres con noecho(), y se oculta el cursor con curs_set(FALSE). Dentro de un bucle infinito, se limpia la pantalla con clear(), se dibuja la pelotita en la posición actual (y, x) con mvprintw(), y se actualiza la pantalla con refresh(). Luego, se pausa brevemente con usleep(DELAY) para controlar la velocidad de movimiento. La variable direction indica si la pelota se mueve hacia la derecha (1) o hacia la izquierda (-1), y se invierte cuando alcanza los límites establecidos por max_x o o. Así, la pelota rebota continuamente de un lado al otro. El programa termina con endwin(), aunque como no hay condición de salida, se debe cerrar manualmente. Este ejemplo es la base para desarrollar juegos con movimiento automático, como el clásico Pong.

aventura.c

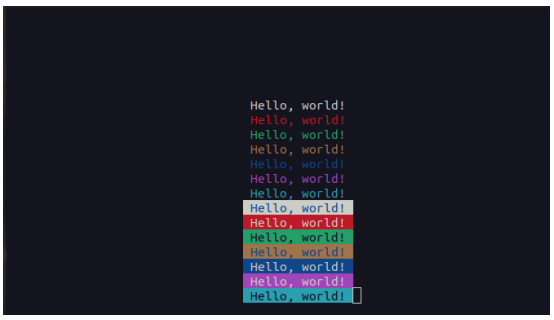
Este programa utiliza la biblioteca ncurses para crear un juego de aventura simple en la terminal, donde el jugador se mueve por un mapa compuesto de césped, montañas y agua. Al iniciar, se configura el entorno con funciones como initscr(), keypad(), y noecho(), y se habilitan los colores con start_color(). El jugador comienza en la esquina inferior izquierda y se mueve por el mapa usando las teclas de dirección o WASD. Cada terreno tiene un color asignado, y el jugador solo puede moverse por las casillas de césped o vacías, controlado por la función is_move_okay(). El mapa se dibuja con diferentes tipos de terrenos, y el movimiento se actualiza en cada iteración del bucle principal. El programa continúa hasta que el jugador presiona 'q' para salir, momento en el que se cierra correctamente la ventana de ncurses con endwin().





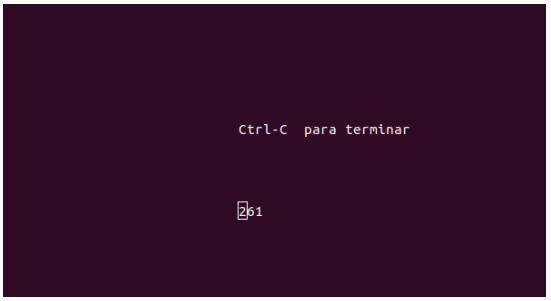
colores.c

Este programa utiliza ncurses para mostrar un mensaje "Hello, world!" en la terminal con colores. Al iniciar, se configura el entorno de ncurses con initscr(), y si la inicialización es exitosa, se activan los colores con start_color(). A continuación, se verifica si la terminal soporta colores y si hay suficientes combinaciones de colores disponibles, en este caso 13. Si es así, el programa define 13 pares de colores usando init_pair(), donde cada par asocia un color de primer plano y uno de fondo. Luego, se imprime el mensaje "Hello, world!" en diferentes colores, moviéndolo a distintas líneas utilizando mvaddstr(). Después de imprimir los mensajes, la pantalla se actualiza con refresh() y se espera 3 segundos para mostrar el efecto de pantalla completa. Finalmente, el programa limpia y cierra la ventana de ncurses con delwin() y endwin().



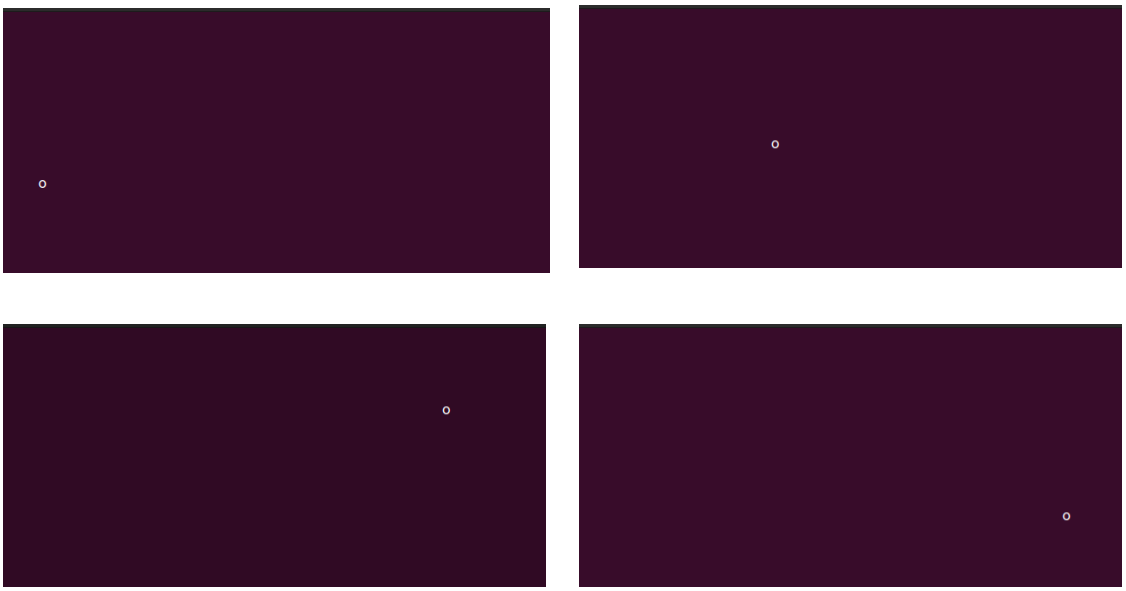
pruncurses.c

Este programa utiliza ncurses para crear una ventana en la terminal que muestra los códigos de teclas presionadas por el usuario. Al iniciar, se configura la pantalla con initscr() y se borra cualquier contenido previo con clear(). Luego, se habilita el modo de entrada sin eco (noecho()) y se activa el procesamiento inmediato de teclas con cbreak(). Se habilita también el uso de teclas especiales, como las teclas de función o las flechas, con keypad(). Se imprime el mensaje "Ctrl-C para terminar" en la posición (7, 30) y, a continuación, el programa entra en un bucle infinito. En cada iteración, el programa captura el código de la tecla presionada con getch() y lo muestra en la posición (12, 30) de la ventana. La pantalla se actualiza constantemente con refresh(), y el programa se termina cuando el usuario presiona Ctrl-C. Al finalizar, endwin() se utiliza para limpiar la pantalla y restaurar el estado original de la terminal.



rebota.c

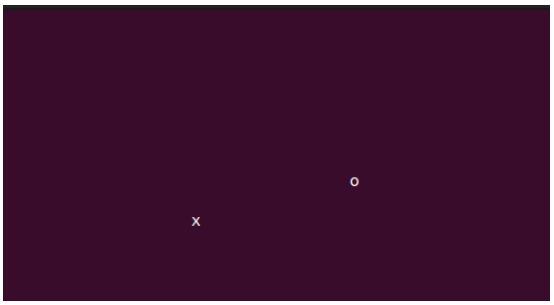
Este programa en C usa la biblioteca ncurses para crear una animación simple de una pelotita ('o') que rebota dentro de la terminal. Al principio, el programa configura la pantalla con initscr() y desactiva el cursor con curs_set(FALSE). Luego, entra en un bucle infinito en el cual se dibuja la pelotita en la posición (x, y) y se actualiza la pantalla con refresh(). La pelotita se mueve en ambas direcciones, horizontal y vertical, mediante las variables directionx y directiony, que controlan el sentido del movimiento. Cada vez que la pelotita alcanza el borde de la pantalla (determinado por las variables max_x y max_y), su dirección se invierte, haciendo que rebote. Para controlar la velocidad del movimiento, se utiliza usleep(DELAY), que pausa el programa por un tiempo determinado entre cada actualización de la pantalla. El bucle continúa indefinidamente hasta que el programa se cierra, momento en el que se llama a endwin() para restaurar el estado original de la terminal.

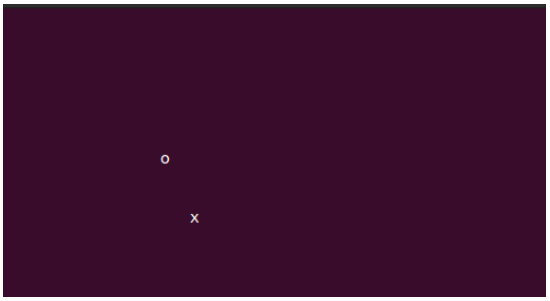


rebota2.c

Este programa utiliza ncurses para crear una animación de una pelotita ('o') que rebota por la pantalla mientras otra figura ('x') se mueve en la dirección horizontal según las teclas presionadas. Al inicio, se configura la pantalla con initscr() y se desactiva el cursor con curs_set(FALSE). En el bucle principal, el programa dibuja dos objetos: una pelotita en la posición (x, y) que rebota dentro de los límites de la ventana (definidos por max_x y max_y), y una figura 'x' en la posición (xc, yc) que se mueve horizontalmente. El movimiento de la 'x' se controla con las teclas 'o' y 'p', donde 'o' mueve la figura a la izquierda y 'p' a la derecha. Para garantizar que la pantalla se actualice sin esperar la entrada de teclado, se usa nodelay(stdscr, TRUE), lo que permite que el programa continúe ejecutándose sin bloquearse. La velocidad de la animación se controla con usleep(DELAY), que pausa el programa brevemente

entre cada actualización de pantalla. El bucle continúa indefinidamente hasta que se cierre el programa, y al final, se llama a endwin() para restaurar el estado de la terminal.





4. Juego sencillo tipo “pong”

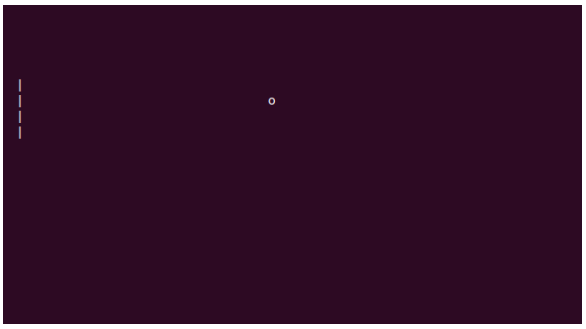
```c
#include <ncurses.h>
#include <unistd.h>

#define DELAY 30000
#define PADDLE_HEIGHT 4
#define PADDLE_X 2

int main() {
    int max_y, max_x;
    int ball_x = 10, ball_y = 10;
    int ball_dx = 1, ball_dy = 1;

    int paddle_y = 5;
    int ch;

    initscr();
    noecho();
    curs_set(FALSE);
    nodelay(stdscr, TRUE); // No bloquea con getch()
    keypad(stdscr, TRUE);  // Habilita flechas

    getmaxyx(stdscr, max_y, max_x);

    while (1) {
        clear();

        // Dibujar la paleta
        for (int i = 0; i < PADDLE_HEIGHT; i++) {
            mvprintw(paddle_y + i, PADDLE_X, "|");
        }


        // Mover la pelota
        ball_x += ball_dx;
        ball_y += ball_dy;

        // Rebote en bordes verticales
        if (ball_y <= 0 || ball_y >= max_y - 1)
            ball_dy *= -1;

        // Rebote en el borde derecho
        if (ball_x >= max_x - 1)
            ball_dx *= -1;

        // Verificar colisión con paleta
        if (ball_x == PADDLE_X + 1) {
            if (ball_y >= paddle_y && ball_y < paddle_y + PADDLE_HEIGHT) {
                ball_dx *= -1; // Rebota
            } else if (ball_x <= 0) {
                break; // La pelota pasó la paleta: game over
            }
        }

        // Leer teclas (w y s para mover paleta)
        ch = getch();
        if (ch == 'w' && paddle_y > 0) {
            paddle_y--;
        } else if (ch == 's' && paddle_y + PADDLE_HEIGHT < max_y) {
            paddle_y++;
        } else if (ch == 'q') {
            break; // Salir manualmente
        }
    }

    endwin();
    return 0;
}
```

Este programa utiliza ncurses para crear un juego simple inspirado en Pong, donde una pelota rebota en la pantalla y el jugador controla una paleta verticalmente para evitar que la pelota se salga de la pantalla. Al iniciar, se configura la pantalla con initscr(), y se desactiva el cursor con curs_set(FALSE). La paleta se dibuja como una serie de caracteres verticales (con |) en una posición fija en el lado izquierdo de la pantalla, y la pelota se dibuja como un 'o'. En cada iteración del bucle principal, la pelota se mueve en función de su velocidad (ball_dx, ball_dy) y rebota en los bordes superior, inferior y derecho de la pantalla. Además, si la pelota colisiona con la paleta, su dirección horizontal se invierte (rebota). El jugador puede mover la paleta usando las teclas 'w' (arriba) y 's' (abajo). Si la pelota pasa la paleta y alcanza el borde izquierdo de la pantalla, el juego termina. El programa también permite salir manualmente con la tecla 'q'. La velocidad del juego se controla con usleep(DELAY). Al final, el programa limpia la pantalla con endwin().

Se ha compilado y ejecutado de esta forma:

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc pong.c -o pong -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./pong
```

Como resultado de la ejecución tenemos lo siguiente:



5. Pantalla de bienvenida

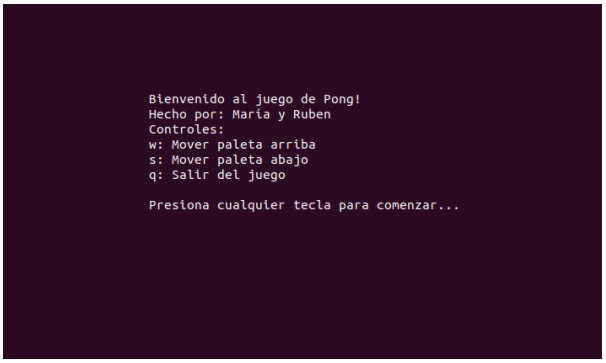
```c
void show_welcome_screen(int max_y, int max_x) {
    clear();
    mvprintw(max_y / 4, max_x / 4, "Bienvenido al juego de Pong!");
    mvprintw(max_y / 4 + 1, max_x / 4, "Hecho por: María y Ruben");
    mvprintw(max_y / 4 + 2, max_x / 4, "Controles:");
    mvprintw(max_y / 4 + 3, max_x / 4, "w: Mover paleta arriba");
    mvprintw(max_y / 4 + 4, max_x / 4, "s: Mover paleta abajo");
    mvprintw(max_y / 4 + 5, max_x / 4, "q: Salir del juego");
    mvprintw(max_y / 4 + 7, max_x / 4, "Presiona cualquier tecla para comenzar...");

    refresh();
    getch();
}
```

La función show_welcome_screen() se encarga de mostrar una pantalla de bienvenida al iniciar el juego. Primero, limpia la pantalla con clear() para asegurarse de que no haya elementos previos. Luego, utiliza mvprintw() para imprimir varios mensajes centrados en la pantalla, incluyendo el título del juego, los créditos de los creadores y las instrucciones de los controles (cómo mover la paleta con 'w' y 's', y cómo salir con 'q', esta funcionará una vez se pase a la pantalla de juego). Finalmente, muestra un mensaje indicando que el usuario debe presionar cualquier tecla para comenzar el juego. Después de imprimir todo el contenido, la función espera la pulsación de una tecla mediante getch(), lo que permite que la pantalla de bienvenida se mantenga visible hasta que el usuario interactúe, momento en el cual el juego comenzará.

Se ha compilado y ejecutado de esta forma:

```
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ gcc pongPantallaInicio.c -o pongInicio -lncurses
maria@maria-Lenovo-IdeaPad-S340-15IIL:~/Escritorio/PDIH/Prácticas/P2/P2-ejemplos/P2-ejemplos$ ./pongInicio
```

Como resultado de la ejecución tenemos lo siguiente:



6. Pantalla final

Se ha compilado y ejecutado de esta forma:

```
ruben-ht@ubuntu-rht:~/Escritorio/PDIH/P2-ejemplos$ gcc pongFinal.c -o pongFinal
-lncurses
ruben-ht@ubuntu-rht:~/Escritorio/PDIH/P2-ejemplos$ ./pongFinal
```

Para empezar, al tener que mostrar al jugador que ha ganado, debemos entender que pueden jugar dos personas de forma simultánea. Por ello, creamos una paleta en la parte derecha de la pantalla y, para que no sea a un único juego, hacemos que cada partida haya un total de 3 puntos hasta su finalización.

```
#define PADDLE2_X 76
#define MAX_GAMES 3
```

Para que ambos jugadores puedan jugar de forma simultánea, hacemos que la paleta derecha use comandos de movimiento diferentes a la paleta izquierda, indicando dicho movimiento en la pantalla de inicio mediante la modificación de show_welcome_screen():

```
void show_welcome_screen(int max_y, int max_x) {
    clear();
    mvprintw(max_y / 4, max_x / 4, "Bienvenido al juego de Pong!");
    mvprintw(max_y / 4 + 1, max_x / 4, "Hecho por: Maria y Ruben");
    mvprintw(max_y / 4 + 2, max_x / 4, "Controles:");
    mvprintw(max_y / 4 + 3, max_x / 4, "w: Mover paleta izquierda arriba");
    mvprintw(max_y / 4 + 4, max_x / 4, "s: Mover paleta izquierda abajo");
    mvprintw(max_y / 4 + 5, max_x / 4, "i: Mover paleta derecha arriba");
    mvprintw(max_y / 4 + 6, max_x / 4, "k: Mover paleta derecha abajo");
    mvprintw(max_y / 4 + 7, max_x / 4, "q: Salir del juego");
    mvprintw(max_y / 4 + 9, max_x / 4, "Presiona cualquier tecla para comenzar...");

    refresh();
    getch();  // Espera una tecla para continuar
}
```

A continuación, creamos la función show_game_result() la cual, tras cada 3 puntos, se ejecutará para mostrar la puntuación de cada jugador, dar la felicitación al ganador y dar las indicaciones para jugar de nuevo o abandonar la partida (añadimos la opción de cambiar la felicitación por “es un empate” por si se modifica el número de puntos de cada partida a un número par, al existir la posibilidad de que hubiese un empate):

```
void show_game_result(int max_y, int max_x, int score1, int score2) {
    clear();
    mvprintw(max_y / 4, max_x / 4, "¡Juego Terminado!");
    mvprintw(max_y / 4 + 1, max_x / 4, "Marcador Final:");
    mvprintw(max_y / 4 + 2, max_x / 4, "Jugador 1: %d", score1);
    mvprintw(max_y / 4 + 3, max_x / 4, "Jugador 2: %d", score2);

    if (score1 > score2) {
        mvprintw(max_y / 4 + 4, max_x / 4, "¡Felicidades, Jugador 1! Has ganado.");
    } else if (score2 > score1) {
        mvprintw(max_y / 4 + 4, max_x / 4, "¡Felicidades, Jugador 2! Has ganado.");
    } else {
        mvprintw(max_y / 4 + 4, max_x / 4, "¡Es un empate!");
    }

    mvprintw(max_y / 4 + 6, max_x / 4, "Presiona 'r' para jugar de nuevo o 'q' para salir...");

    refresh();
}
```

Para empezar, hacemos que tras cada punto se sume el contador del jugador que ha marcado. Además, para hacer el juego más equilibrado, hacemos que la pelota vaya en dirección al jugador que ha anotado punto. Es decir, si ha anotado punto el jugador de la derecha (ha anotado en el lado izquierdo de la pantalla), la pelota se dirija a la paleta de la derecha, ídem si anota el jugador de la izquierda:

```
if (ball_x >= max_x - 1) {
    score1++;
    ball_dx = -1;
    break; // Salir del ciclo de juego
}

if (ball_x <= 0) {
    score2++;
    ball_dx = 1;
    break; // Salir del ciclo de juego
}
```

Además, creamos la función reset_game() que se encarga de poner los marcadores y el número de puntos jugados a cero (ya que tras cada punto se incrementará en uno) para iniciar una nueva partida en caso de que los jugadores indiquen que quieren volver a jugar:

```
void reset_game(int* score1, int* score2, int* game_count) {
    *score1 = 0;
    *score2 = 0;
    *game_count = 0;
}
```

Para que pueda haber un reinicio, hacemos que si se pulsa la tecla “r”, se lleve a cabo el reseteo de la función reset_game, de tal manera que los marcadores pasen a o y, más importante, también se ponga a o el conteo de puntos jugados de tal forma que se vuelva a jugar de nuevo como si fuese una nueva partida al repetirse el bucle:

```
show_game_result(max_y, max_x, score1, score2);

while (1) {
    ch = getch();
    if (ch == 'r') {
        reset_game(&score1, &score2, &game_count);
        break;
    } else if (ch == 'q') {
        break;
    }
}
```

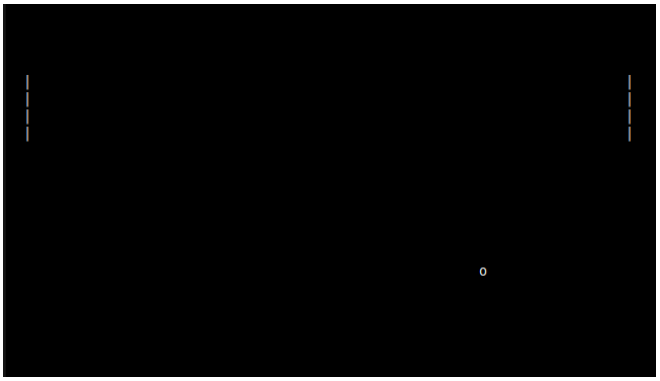
Por último, indicar que en cada reinicio, se estipula que la pelota siempre se dirija abajo a la derecha en el primer punto, independientemente de quién haya anotado el primer punto:

```
while (game_count < MAX_GAMES) {

    ball_x = max_x / 2;
    ball_y = max_y / 2;


    if(game_count == 0){
    ball_dx = 1;
    }
    ball_dy = 1;
```

Por ello, el programa se verá de la siguiente forma:
Para empezar, se muestra la pantalla con los controles para cada jugador:

```
Bienvenido al juego de Pong!
Hecho por: María y Ruben
Controles:
w: Mover paleta izquierda arriba
s: Mover paleta izquierda abajo
i: Mover paleta derecha arriba
k: Mover paleta derecha abajo
q: Salir del juego

Presiona cualquier tecla para comenzar...
```

A continuación, se juega la partida, manejando cada jugador una paleta:



Finalmente, se muestra la pantalla de fin, mostrando el marcador y felicitando al ganador, además de decir dar la opción de jugar de nuevo o salir del juego: