

PDIH - UGR

Práctica 2 - Ángel Gómez Ferrer

Índice

| | |
|--|----------|
| Práctica 2 - Ángel Gómez Ferrer | 1 |
| Índice..... | 1 |
| Instalación..... | 1 |
| Ejemplos..... | 2 |
| Ejemplo hello..... | 2 |
| Ejemplo ventana..... | 2 |
| Ejemplo Pelotita..... | 3 |
| Ejemplo rebota..... | 3 |
| Ejemplo rebota2..... | 4 |
| Juego Pong..... | 4 |
| Funcionamiento..... | 4 |
| Ventana de bienvenida..... | 4 |
| Partida..... | 6 |

Instalación

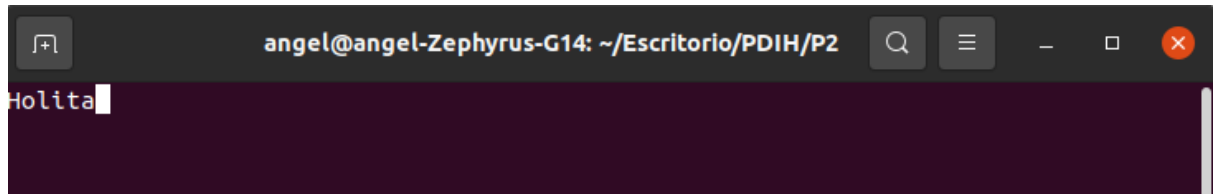
Para la instalación simplemente ejecutamos el comando que se puede ver en la consola, y ya tendríamos lo necesario para compilar haciendo uso de la librería ncurses:

```
angel@angel-Zephyrus-G14:~/Escritorio/PDIH$ sudo apt-get install libncurses5-dev libncursesw5-dev
[sudo] contraseña para angel:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.
  linux-headers-5.13.0-28-generic linux-hwe-5.13-headers-5.13.0-28
  linux-image-5.13.0-28-generic linux-modules-5.13.0-28-generic
  linux-modules-extra-5.13.0-28-generic
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 507 no actualizados.
Se necesita descargar 1.956 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 focal/main amd64 libncurses5-dev amd64 6.2-0ubuntu2 [976 B]
Des:2 http://es.archive.ubuntu.com/ubuntu focal/main amd64 libncursesw5-dev amd64 6.2-0ubuntu2 [980 B]
Descargados 1.956 B en 0s (4.062 B/s)
Seleccionando el paquete libncurses5-dev:amd64 previamente no seleccionado.
(Leyendo la base de datos ... 449388 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../libncurses5-dev_6.2-0ubuntu2_amd64.deb ...
Desempaquetando libncurses5-dev:amd64 (6.2-0ubuntu2) ...
Seleccionando el paquete libncursesw5-dev:amd64 previamente no seleccionado.
Preparando para desempaquetar .../libncursesw5-dev_6.2-0ubuntu2_amd64.deb ...
Desempaquetando libncursesw5-dev:amd64 (6.2-0ubuntu2) ...
Configurando libncurses5-dev:amd64 (6.2-0ubuntu2) ...
Configurando libncursesw5-dev:amd64 (6.2-0ubuntu2) ...
angel@angel-Zephyrus-G14:~/Escritorio/PDIH$
```

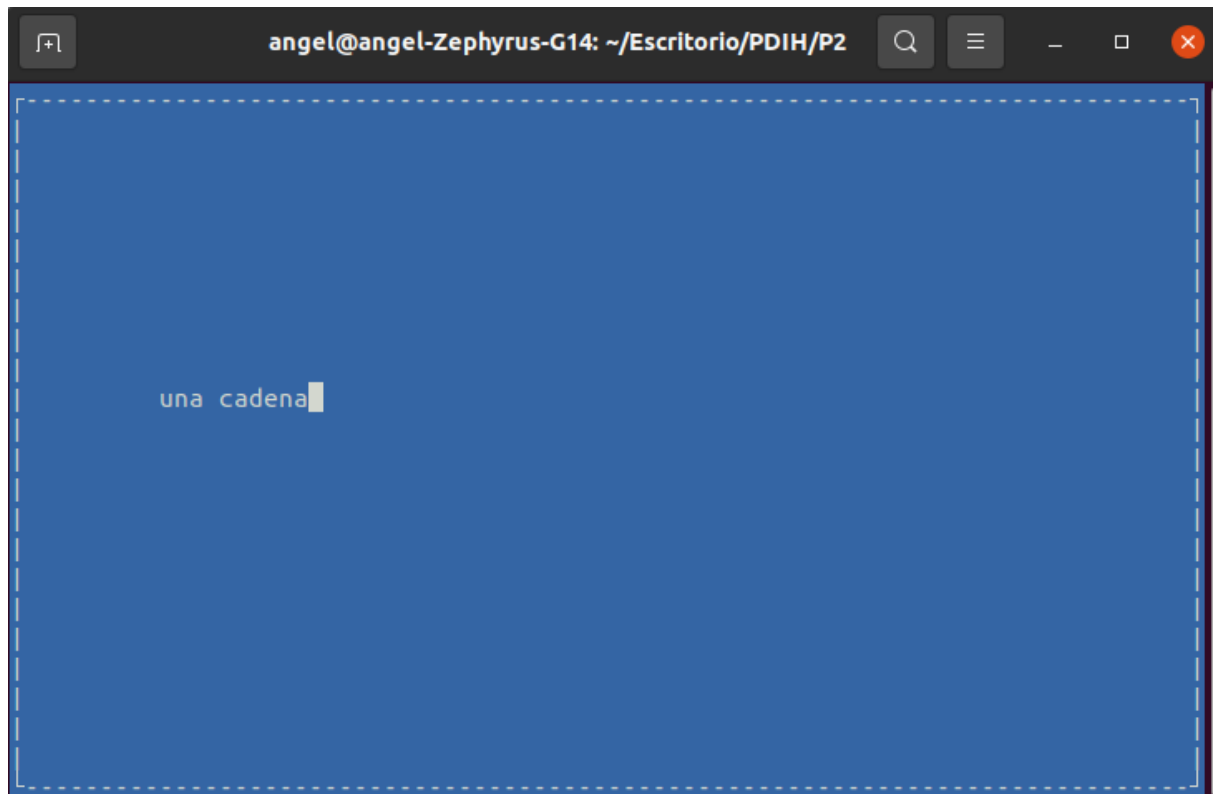
Ejemplos

Compilamos: con el siguiente comando: `gcc hello.c -o hello -lncurses`
Y ejecutamos el primer ejemplo: `./hello`

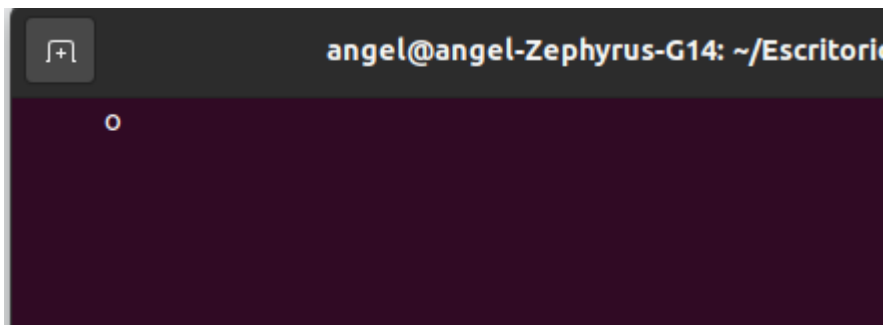
Ejemplo hello

A terminal window with a dark background. The title bar shows the user 'angel' on a machine named 'angel-Zephyrus-G14' at the directory '~/Escritorio/PDIH/P2'. The terminal displays the text 'Holita' followed by a white cursor.

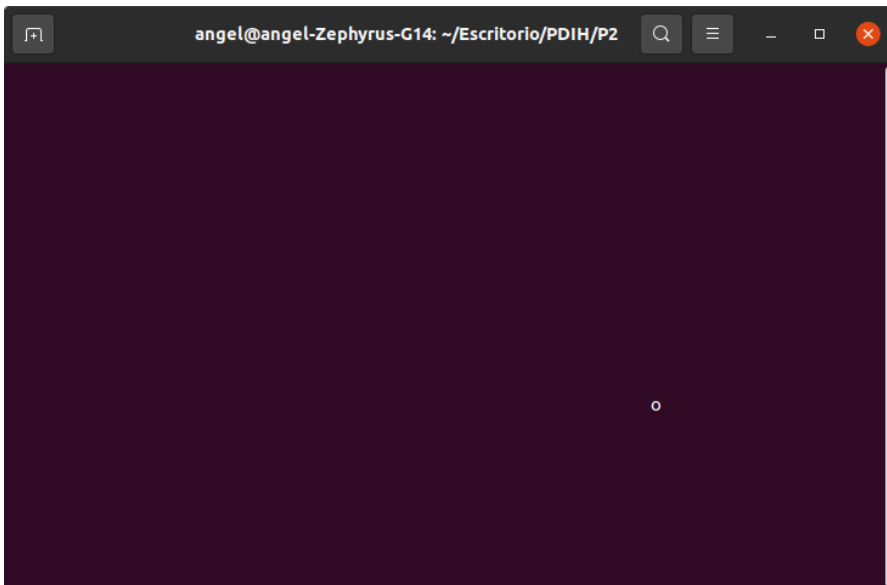
Ejemplo ventana

A terminal window with a dark background. The title bar shows the user 'angel' on a machine named 'angel-Zephyrus-G14' at the directory '~/Escritorio/PDIH/P2'. The terminal displays a large blue rectangular area with a dashed white border. Inside this area, the text 'una cadena' is followed by a white cursor.

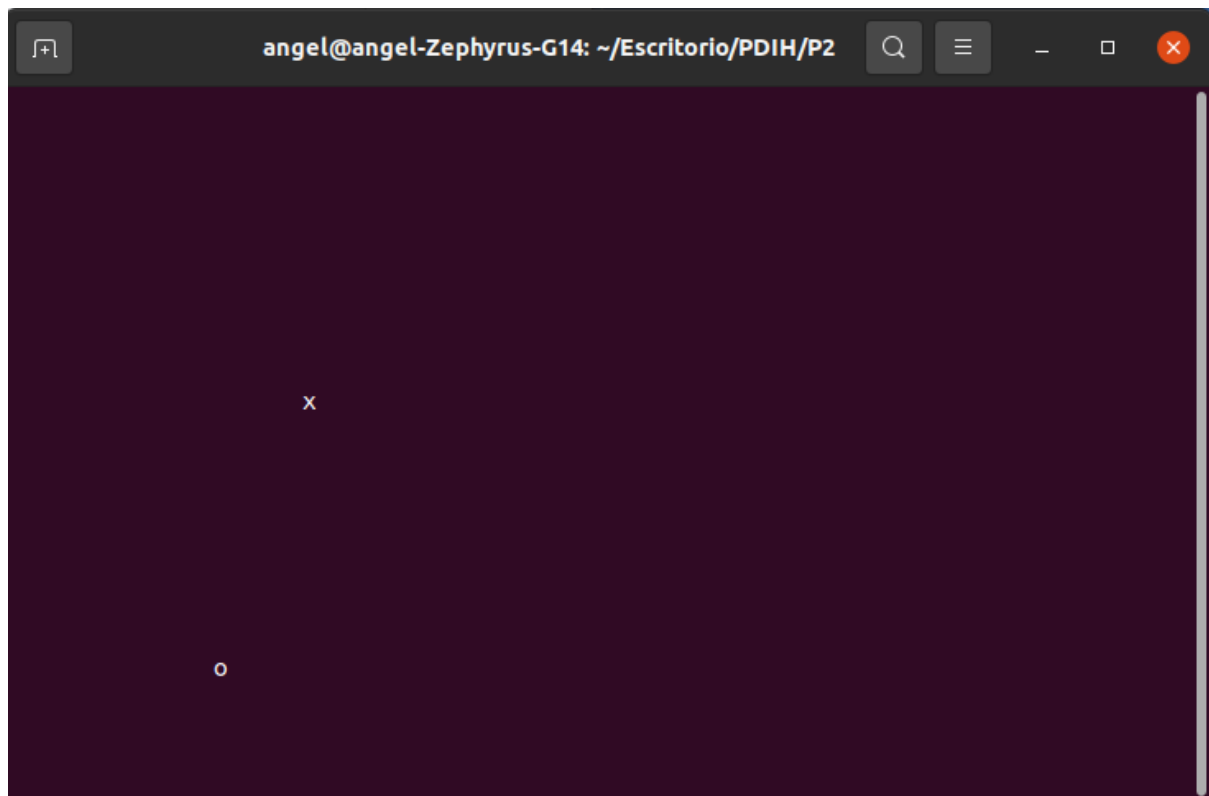
Ejemplo Pelotita



Ejemplo rebota



Ejemplo rebota2



Juego Pong

Funcionamiento

En este apartado iré explicando el código por partes en el orden que suceden las cosas e iré mostrando los resultados que sean necesarios.

Ventana de bienvenida

Esta será la primera ventana que veamos al iniciar, nos mostrará las instrucciones del juego y mis datos.

Para ello iniciaremos unos cuantos pares de colores (en mi caso acabé usando letras azules y fondo blanco el par 1).

Tras esto utilizaremos una función importante para la realización del juego `getmaxyx(...)` con la que guardaremos las variables `screen_y` y `screen_x`, es decir el tamaño de la ventana de la terminal en la que se ejecutará.

Crearemos una ventana `newwin(...)` y le asociaremos el color elegido con `wbkgd(window, COLOR_PAIR(1))`

También hemos creado un recuadro alrededor del interior de la ventana con `box`. Tras esto he introducido con `mvwprintw` todas las instrucciones y mis datos.

Y por último obtendremos un carácter para dar comienzo a la partida.

```
20 if (has_colors() == FALSE) {
21     endwin();
22     printf( format: "Your terminal does not support color\n");
23     exit( status: 1);
24 }
25
26 start_color();
27 init_pair(1, COLOR_BLUE, COLOR_WHITE);
28 init_pair(2, COLOR_RED, COLOR_WHITE);
29 init_pair(3, COLOR_WHITE, COLOR_BLUE);
30 clear();
31
32 refresh();
33 getmaxyx(stdscr, screen_y, screen_x);
34
35 WINDOW *window = newwin(screen_y, screen_x, 0, 0);
36 wbkgd(window, COLOR_PAIR(1));
37 box(window, '|', '-');
38
39 mvwprintw(window, 1, 1, "Hecho por: Angel Gomez Ferrer (Github: https://github.com/Angelgf22)");
40 mvwprintw(window, 4, 4, "Controles:");
41 mvwprintw(window, 5, 4, "Jugador izquierda:");
42 mvwprintw(window, 6, 4, "\t- W : Subir");
43 mvwprintw(window, 7, 4, "\t- S : Bajar");
44 mvwprintw(window, 8, 4, "Jugador derecha:");
45 mvwprintw(window, 9, 4, "\t- KEY_UP : Subir");
46 mvwprintw(window, 10, 4, "\t- KEY_DOWN : Bajar");
47 mvwprintw(window, 11, 4, "El primer jugador en llega a 10 goles gana!");
48 mvwprintw(window, 12, 4, "Pulse una tecla para comenzar el juego, utiliza la tecla Q para cerrar.");
49 mvwprintw(window, 13, 4, "Suerte!");
50 wrefresh(window);
51
52 getch();
53 endwin();
```

```
angel@angel-Zephyrus-G14: ~/Escritorio/PDIH/P2
Hecho por: Angel Gomez Ferrer (Github: https://github.com/Angelgf22)

Controles:
Jugador izquierda:
- W : Subir
- S : Bajar
Jugador derecha:
- KEY_UP : Subir
- KEY_DOWN : Bajar
El primer jugador en llega a 10 goles gana!
Pulse una tecla para comenzar el juego, utiliza la tecla Q para cerrar.
Suerte!
```

Partida

Para las barras he creado una estructura con la siguiente forma:

```
typedef struct{int x, y, goals, size;} Paddle;
```

Cada barra tendrá su posición x, y, los goles marcados y el tamaño que tendrá.

Primero inicializamos las configuraciones básicas y los valores de las barras que he llamado `right_paddle` y `left_paddle`:

```
initscr();
noecho();
curs_set(false);
nodelay(stdscr, true);
keypad(stdscr, true);

right_paddle.size = 7;
right_paddle.x = screen_x - 2;
right_paddle.y = screen_y/2 - right_paddle.size/2;
right_paddle.goals = 0;

left_paddle.size = 7;
left_paddle.x = 2;
left_paddle.y = screen_y/2 - left_paddle.size/2;
left_paddle.goals = 0;
```

Tras esto creamos el bucle para iterar los pasos de la partida.

```

while(!quit) {
    clear();
    mvprintw(y, x, "o");
    for(i = 0 ; i < screen_y; ++i) {
        mvprintw(i, screen_x/2, "|");
    }

    for(i = 0 ; i < right_paddle.size; ++i) {
        mvprintw(right_paddle.y + i, right_paddle.x, "|");
    }

    for(i = 0 ; i < left_paddle.size; ++i) {
        mvprintw(left_paddle.y + i, left_paddle.x , "|");
    }
    mvprintw(0, screen_x/4 , "%i", left_paddle.goals);
    mvprintw(0, screen_x- screen_x/4 , "%i", right_paddle.goals);

    switch(getch()){
        case 'w':
            if(left_paddle.y > 0)
                left_paddle.y--;
            break;
        case 's':
            if(left_paddle.y+left_paddle.size < screen_y)
                left_paddle.y++;
            break;
        case KEY_UP:
            if(right_paddle.y > 0)
                right_paddle.y--;
            break;
        case KEY_DOWN:
            if(right_paddle.y+right_paddle.size < screen_y)
                right_paddle.y++;
            break;
        case 'q':
            quit = true;
            break;
    }

    refresh();
}

```

En esta primera parte del bucle, crearemos la pelota, pintaremos las barras derecha e izquierda con dos bucles for y conseguiremos el carácter de movimiento de cada uno de los jugadores, también detectaremos si se quiere terminar la partida con la tecla q. He aplicado algunos ajustes en las coordenadas de las barras para que se ajusten al tamaño que tenga y pueda ser variable, además cada vez que se vayan a mover comprobaremos que la barra no esté en los límites de 'y' de la pantalla.

```

usleep(DELAY);

next_x = x + direction_x;
next_y = y + direction_y;

//Colisionan (right)
if(right_paddle.x == x){
    for (i = 0; i < right_paddle.size; ++i) {
        if((right_paddle.y + i) == y){
            direction_x*= -1;
            x+= direction_x;
        }
    }
}

```

```

//Colisionan (left)
if(left_paddle.x == x){
    for (i = 0; i < left_paddle.size; ++i) {
        if((left_paddle.y + i) == y){
            direction_x*= -1;
            x+= direction_x;
        }
    }
}

if (next_x >= screen_x) {
    left_paddle.goals++;
    x = screen_x/4;
    y = screen_y/2;
    direction_y*= -1;
}else if (next_x < 0) {
    right_paddle.goals++;
    x = screen_x - screen_x/4;
    y = screen_y/2;
    direction_y*= -1;

}else {
    x+= direction_x;
}

if (next_y >= screen_y || next_y < 0) {
    direction_y*= -1;
} else {
    y+= direction_y;
}

if(left_paddle.goals == 10) {
    quit = true;
}

if(right_paddle.goals == 10) {
    quit = true;
}

}

```

En esta última parte del bucle aplicaremos el delay en mi caso 40000 definido así:

```
#define DELAY 40000
```

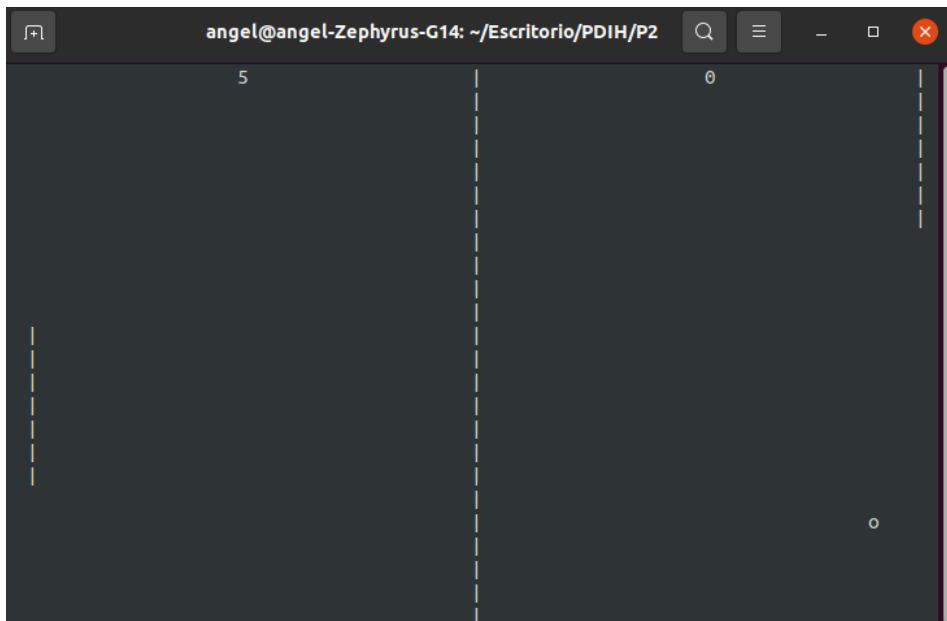
que dará tiempo entre las diferentes iteraciones del bucle.

Crearemos el movimiento de la pelota dándole una dirección la cual se le sumará para que avance en tal dirección.

Crearemos las colisiones con las dos barras, y sumaremos goles en el caso de que la pelota toque los extremos de la pantalla en 'x', aplicando también las nuevas posiciones para la pelota.

Por último comprobaremos las colisiones de la pelota en caso de chocar con los bordes en el eje 'y' y verificaremos que ninguno de los jugadores haya llegado a 10 en cuyo caso analizaremos la partida terminando así el bucle.

Aquí mostraré capturas de pantalla de la ejecución de una partida:



Con diferentes tamaños:

