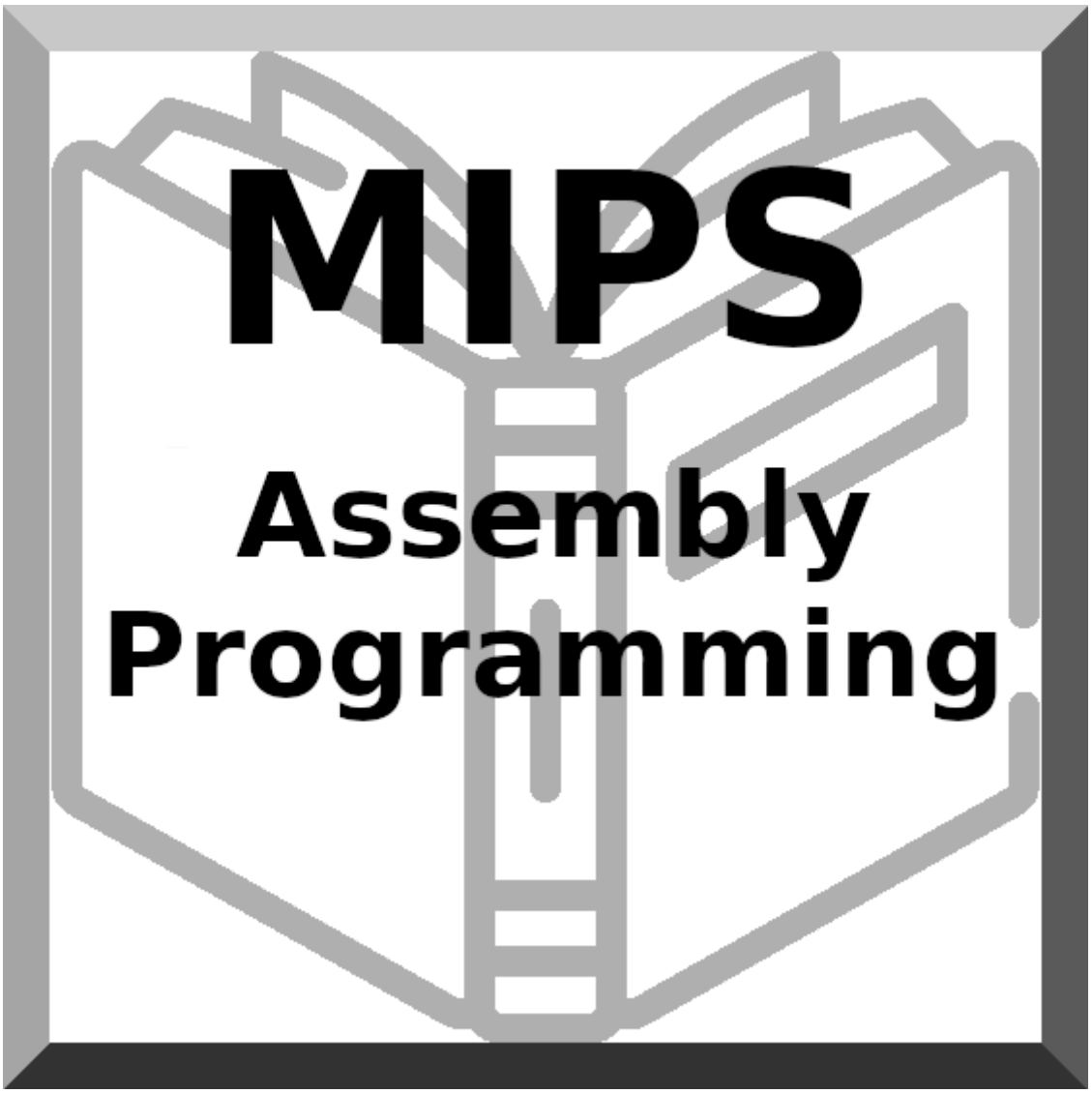
PRÁCTICA 2 ARQUITECTURA GRÁFICA



Carla Calvache Amador, Rubén Pedroso Praena
2º Diseño y Desarrollo de Videojuegos URJC Móstoles

ÍNDICE

1. Introducción

1.1. Objetivo del proyecto
El objetivo principal de este proyecto es desarrollar un videojuego sencilloa través de MIPS, haciéndonos aplicar lo aprendido en clase para demostrar nuestros conocimientos. Además, se pretende que nos familiaricemos con el entorno de desarrollo y simulación MIPS.

1.2. Descripción general del videojuego
El videojuego implementado es una versión simplificada del clásico "Pong", pero haciéndolo ver como un campo de fútbol. Consta de dos palas controladas por los jugadores y una pelota que rebota por la pantalla. El objetivo es evitar que la pelota atraviese la portería contraria (el borde). Si uno de los jugadores falla, se considera gol y se resetea la partida.

2. Controles del Juego

2.1. Controles del Jugador 1

- Tecla 'A': Mover la pala hacia la izquierda.
- Tecla 'D': Mover la pala hacia la derecha.

2.2. Controles del Jugador 2

- Tecla 'J': Mover la pala hacia la izquierda.
- Tecla 'L': Mover la pala hacia la derecha.

3. Descripción de la Pantalla de Juego

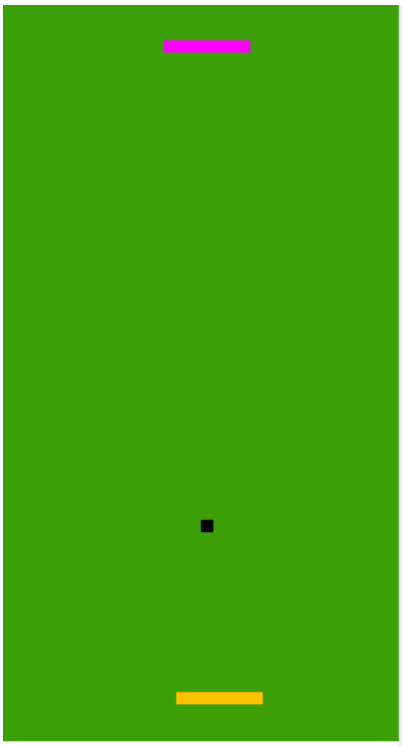
3.1. Elementos visuales

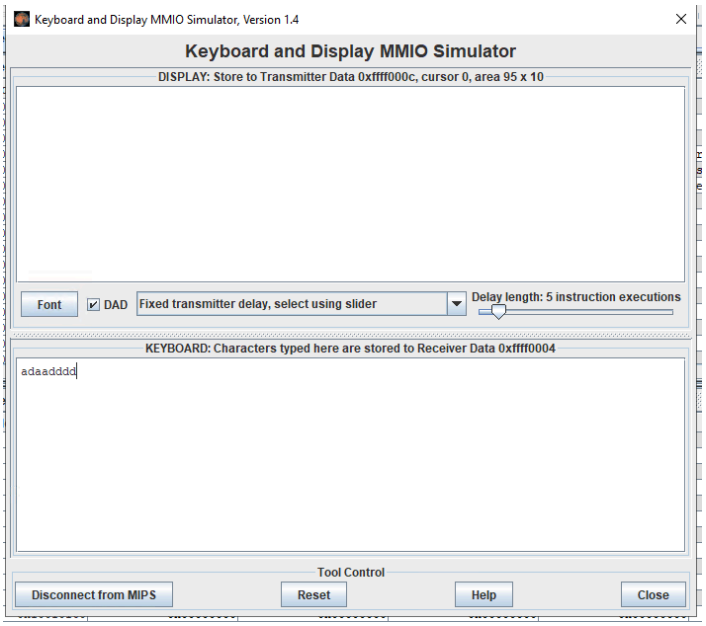
- Pala abajo (Jugador 1)
- Pala arriba (Jugador 2)
- Pelota
- Líneas de límite en los bordes de la pantalla (Para detectar el gol)
- Fondo (no es interactuable pero si se vuelve a pintar cada frame)

3.2. Colores utilizados

- Fondo: Verde
- Palas: La del jugador 1 es naranja, y la del jugador 2 es rosa
- Pelota: Negra
- Bordes: Blancos

3.3. Ejemplos visuales (capturas de pantalla)





4. Funcionamiento Interno del Código

4.1. Fase de inicialización

La fase de inicialización comienza en la etiqueta main, donde se muestra al usuario un mensaje preguntando si quiere comenzar el juego mediante una llamada al sistema (syscall v0 = 50). Si el usuario acepta (valor de $a0 distinto de cero), se salta a la etiqueta init_game.

En init_game se realiza el dibujado inicial del fondo mediante la subrutina draw_background, que recorre toda la memoria del display (0x10008000) escribiendo bloques del color definido como fondo (verde). Esto prepara el escenario gráfico sobre el que se desarrollará el juego.

4.2. Bucle principal del juego

Después de la inicialización, el programa entra en un bucle infinito representado por la etiqueta update. Dentro de este bucle se hace lo siguiente en cada iteración:

- Se espera un pequeño retardo (syscall 32) para permitir que el usuario perciba los cambios visuales.
- Se actualiza el estado del juego:
    - Movimiento de las palas (movimiento_j1 y movimiento_j2)
    - Movimiento de la pelota (movimiento_pelota)
- Se redibujan todos los elementos:

    - Bordes del campo (draw_top)
    - Pala del jugador 1 (j1)
    - Pala del jugador 2 (j2)
    - Pelota (pelota)
- Se comprueban las colisiones (col_j)
- Se vuelve a update, repitiendo el ciclo.

4.3. Dibujo de elementos

El sistema gráfico se basa en un mapa de bits donde cada celda es de 8x8 píxeles, representado como una palabra en memoria. Los elementos gráficos se dibujan accediendo directamente a esta memoria:

- Fondo (draw_background): llena todo el display con el color de fondo.
- Bordes superior e inferior (draw_top): dibuja líneas horizontales blancas en la primera y última fila del campo. (no se ocnsideró necesario hacer un draw_bottom, dado a que siempre que pintamos la de arriba vamos a querer pintar la de abajo, por lo que simplemente está una debajo de la otra)
- Pala jugador 1 (j1): se dibuja cerca del borde inferior, con un color amarillo. Su posición vertical se basa en la variable pos_j1.
- Pala jugador 2 (j2): se ubica cerca del borde superior y tiene color rosa. Se mueve verticalmente usando pos_j2.
- Pelota (pelota): se dibuja como un pequeño bloque negro centrado con fondo alrededor, en base a pos_pelota_x y pos_pelota_y.

4.4. Gestión de entrada del teclado

Las entradas del teclado se gestionan leyendo los registros de memoria (0xFFFF0000 y 0xFFFF0004), correspondientes al estado del teclado y a la tecla pulsada.

- Jugador 1 (pala de abajo):
    - Tecla a: mueve la pala hacia la izquierda (reduce pos_j1).
    - Tecla d: mueve la pala hacia la derecha (incrementa pos_j1).
- Jugador 2 (pala de arriba):
    - Tecla j: mueve la pala hacia la izquierda (pos_j2--).
    - Tecla l: mueve la pala hacia la derecha (pos_j2++).

En ambas palas, en caso de llegar al límite, esto se detecta y no se mueven. Cada acción modifica la posición correspondiente si se respeta el rango permitido en pantalla.

4.5. Movimiento de la pelota

La subrutina movimiento_pelota actualiza la posición de la pelota sumando sus velocidades actuales (velocidad_x, velocidad_y) a sus coordenadas.

Luego, se realizan varias comprobaciones:

- Rebote en paredes laterales: Si la pelota alcanza los bordes izquierdo o derecho del campo, se invierte velocidad_x.
- Gol en portería de abajo: Si pos_pelota_y >= 63, se interpreta como un gol en la portería del jugador 1.
    - Se reinicia la posición de la pelota en el centro.
    - Se detiene momentáneamente y se muestra un efecto visual.
    - Además de visual, también hace un efecto sonoro con:
        - $a2 7 (instrumento)
        - $a3 100 (volumen)
        - $a0 40 (frecuencia (hacerlo agudo))
        - $a1 2000 (duración en ms)
- Gol en portería de arriba: Si pos_pelota_y <= 0, el gol es para el jugador 2.
    - Se aplica el mismo tratamiento de reinicio y pausa.

Finalmente, se escriben las nuevas posiciones actualizadas en las variables correspondientes.

4.6. Colisiones y condiciones de victoria

Las colisiones entre la pelota y las palas se detectan en la subrutina col_j:

- Se comprueba si la pelota se encuentra en la misma fila que las palas (líneas 58–59 para J1, y 4–5 para J2).
- Se calcula la distancia entre la posición horizontal de la pelota y la de la pala.
    - Si la distancia es menor o igual que 3, se considera que ha tocado la pala:
        - Se invierte la dirección vertical de la pelota (velocidad_y).
        - Se ajusta la dirección horizontal (velocidad_x) dependiendo de si el rebote fue hacia la izquierda o derecha.
- Si no hay colisión, la pelota continúa su trayectoria hacia la portería, resultando en un gol del jugador contrario.

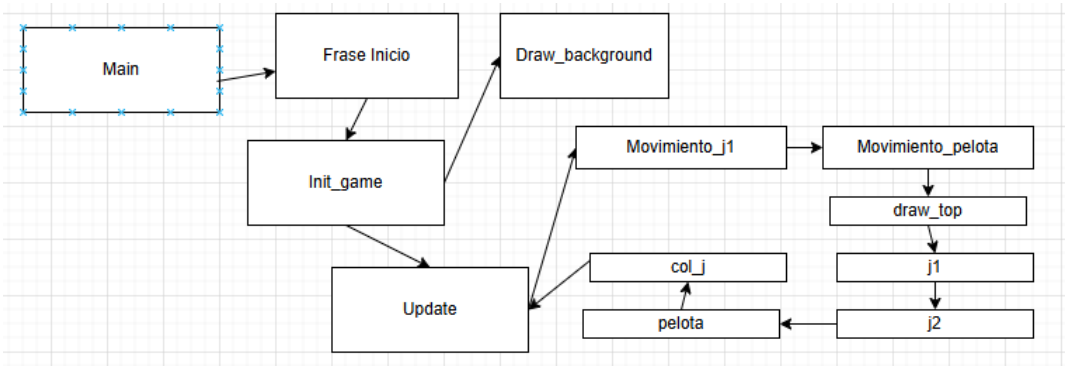
5. Diagrama de Flujo del Programa

5.1. Explicación general del flujo

El programa sigue un ciclo típico de un bucle de juego, con las siguientes fases principales:

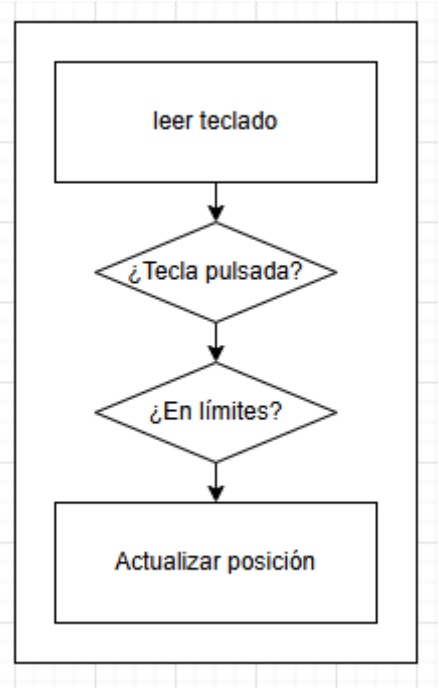
1. *Inicio y Menú:*
    - Se muestra un mensaje (frase_inicio) preguntando al usuario si desea comenzar el juego.
    - Si el usuario acepta, se llama a la subrutina init_game.
2. *Inicialización (init_game):*
    - Se dibuja el fondo del campo mediante la subrutina draw_background.
3. *Bucle principal (update):*
    - En cada iteración del bucle:
        - Se produce una pausa para sincronizar los frames (syscall con código 32).
        - Se actualiza la posición del jugador 1 leyendo el teclado (movimiento_j1). Lo mismo con el jugador 2
        - Se actualiza la posición de la pelota (movimiento_pelota).
        - Se redibujan los elementos principales en pantalla:
            - Líneas superior e inferior (draw_top)
            - Palas de jugadores (j1 y j2)
            - Pelota (pelota)
        - Se comprueban colisiones (col_j), tanto con las palas como con los bordes o porterías.

5.2. Relación entre funciones principales



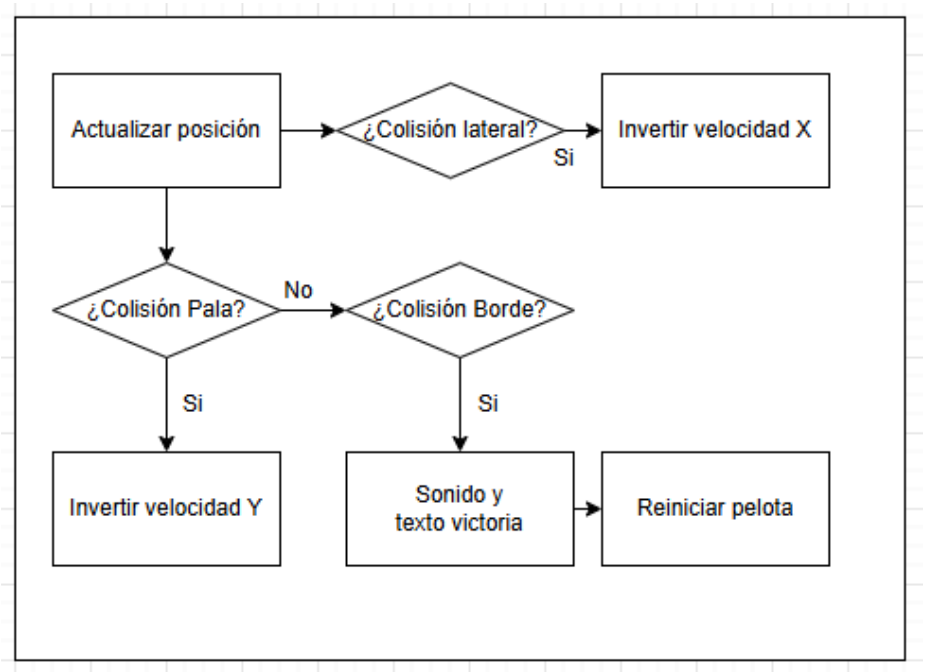
Explicaré ahora cada uno de las funciones dentro de update:

Movimiento_j1:



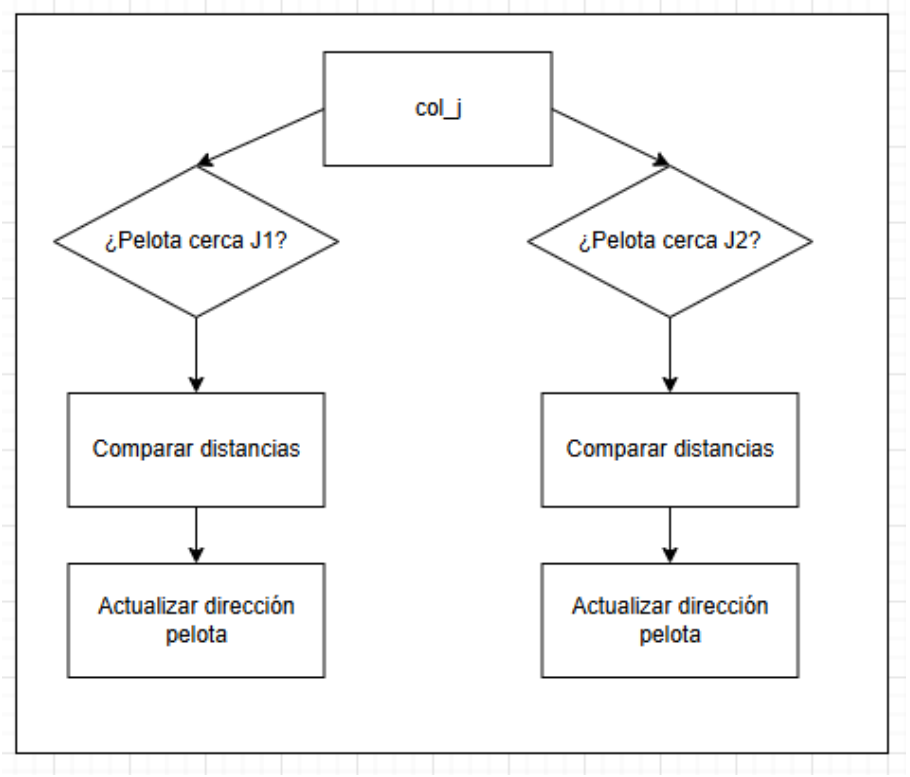
Después de esto, se llama a movimiento_j2 sin llamar de nuevo a la función principal update, por eso no está en el diagrama, pero hace básicamente lo mismo de movimiento_j1.

Movimiento_pelota:



Las funciones draw_top, j1, j2 y pelota solo dibujan de nuevo la pantalla en función de movimiento_j1 y movimiento_pelota, por lo que no es necesario explicarlas con más detalle.

col_j:



6. Ampliaciones Implementadas

- Se añadió una condición de victoria.
- Se mostró un mensaje personalizado en caso de ganar un jugador.
- Se optimizó la gestión de entrada.
- Se consiguió insertar un sonido en caso de gol.