TIC-TAC-TOE

Descripción del proyecto:

Tic-tac-toe se trata sobre el juego en el cual hay que colocar piezas en un tablero de 3x3, buscando alinear 3 piezas antes de que el jugador contrario lo haga. El juego se realiza por turnos. El proyecto se realizó en lenguaje ensamblador MIPS y se programo en MARS. Usando bitmap display como su interfaz gráfica.

Manual de uso:

Primera parte:

1. Se debe de descargar java (última versión) y el archivo para ejecutar MARS.
2. Se debe de descargar y guardar el archivo 3 en raya.asm
3. Después de ejecutar el archivo MARS, abrir el archivo de 3 en raya.asm.
4. Se debe de abrir el bitmap en MARS:
    i. Ir a la opción Tools de MARS
    ii. Seleccionar Bitmap display
    iii. Ajustar los parámetros del tamaño de píxel, ambos en 32 y ajustar los parámetros del display, ambos en 512.
    iv. Oprimir la opción de conectarlo con MIPS
5. Compilar el archivo y ejecutarlo.

Segunda parte:

1. Para jugar se debe de colocar el numero entre 0 y 8, los cuales corresponden a la casilla, en la terminal.
2. Después de colocar el número y presionar enter, la ficha se colocará en la interfaz.
3. Se busca posicionar 3 fichas alineadas.
4. En la terminal se mencionará que jugador tiene el turno.
5. Cuando se intenta colocar una pieza que ya se colocado antes, se volverá a preguntar hasta que ingrese un número valido.
6. Cuando uno de los dos jugadores gane, aparece una opción de reiniciar el juego. Ingrese s si desea reiniciar o n si desea terminar el juego.

División del archivo:

- **main**: Es el punto de entrada del programa. Este inicializa el juego llamando a initTable y llama las funciones correspondientes para crear el tablero. Además, junto con Game realiza la lógica principal del juego, donde pide la entrada al usuario, coloca la ficha, cambia de usuario y revisa el ganador.
- **initTable**: Inicializa el tablero y deja el espacio con 0`s.
- **playerMovement**: Maneja el movimiento del jugador. Verifica si la celda seleccionada está vacía antes de realizar el movimiento. Si la celda está vacía, guarda el número del jugador en la celda correspondiente del tablero. Además, revisa si la entrada está dentro de los limites (0-8) si no vuelve al preguntar.

- **winner**: Verifica si hay un ganador. Verifica si algún jugador ha ganado comprobando las filas, columnas y diagonales. Si encuentra tres celdas consecutivas con el mismo valor, declara al ganador. Además, revisa si hay empate y pregunta si desea reiniciar el juego.
- **userInput:** Solicita y procesa la entrada del usuario. Imprime mensajes solicitando la entrada del usuario y lee el número ingresado. Actualiza el registro $t2 con el valor leído.
- **changePlayer:** Alterna entre los jugadores. Cambia el jugador actual alternando entre el jugador 1 y el jugador 2. Actualiza el color y el número del jugador actual.
- **selected:** Selecciona la celda correspondiente al movimiento. Calcula la posición en el display para la celda seleccionada por el jugador. Ajusta las coordenadas para dibujar el movimiento en la pantalla.
- **drawMovement:** Dibuja el movimiento en la pantalla. Escribe el color correspondiente en la posición calculada del display.
- **createVerticalLine**: Dibuja líneas verticales en el tablero. Dibuja una línea vertical en la posición especificada del display. Repite el proceso hasta completar la línea.
- **createHorizontalLine**: Dibuja líneas horizontales en el tablero. Dibuja una línea horizontal en la posición especificada del display. Repite el proceso hasta completar la línea.
- **replayTable**: Reinicia display a color negro.
- **exit**: Finaliza la ejecución del programa.

Errores:

- No imprime user, pero si imprime lo que sigue.

Mejoras:

- Que imprima user.
- Que el tablero se cree mediante un método createTable
- Tener un contador de partidas ganadas por jugador
- Limpiar el display al finalizar la partida
- Optimizar el uso de registros
- Quitar las duplicaciones de código

Resultados:

Se logra lo que se propuso. Se puede jugar el 3 en raya o Tic-Tac-Toe de forma completa y funcional, se muestran las fichas en el bitmap. Se puede identificar cuando gana y en tal caso reiniciar el juego.

Conclusiones:

Este proyecto fomento en nosotros el conocimiento y el uso de lenguaje ensamblador, en MIPS. Nos impulsó a investigar las formas de interfaces graficas y su uso en MARS. Y consecuentemente estos aprendizajes nos ayudara a mejorar nuestras capacidades de programación y razonamiento para el futuro.

Autores:

Axel Rojas Retana C36944

José Pablo Brenes Coto C31289

Diagramas:

Programa Principal

Inicio

Inicializar table y display

Inicializar los registros:
$t0,0
$t1,1
$t5,4
$t9,0

LLamar a la función InitTable que inicializará un vector con 0, que contendrá las jugadas de los jugadores

Inicialirar los registros:
$t0,0xFFFFFF
$t1,20

LLamar a la función createVerticalLine, para crear la primer línea vertical del tablero

Inicialirar los registros:
$t1,320
$t2,0

LLamar a la función createHorizontallLine, para crear la primer línea horizontal del tablero

Inicialirar los registros:
$t1,40

LLamar a la función createVerticalLine, para crear la segunda línea vertical del tablero

Inicialirar los registros:
$t1,640
$t2,0

LLamar a la función createHorizontallLine, para crear la segunda línea horizontal del tablero

Inicialirar los registros:
$t0,
0x00FF00
$t4,1

Entrar en la función de game, encargada de gestionar el funcionamiento del juego

Inicialirar el registro:
$t3, 20

Llamar a la función userInput, encargada de obtener las jugadas mediante inputs

Llamar a la función playerMovement, encargada de verificar jugadas validas y guardarlas

Llamar a la función selected, encargada de dibujar la jugada en el tablero

Llamar a la función selected, encargada de cargar la posición para dibujar la jugada

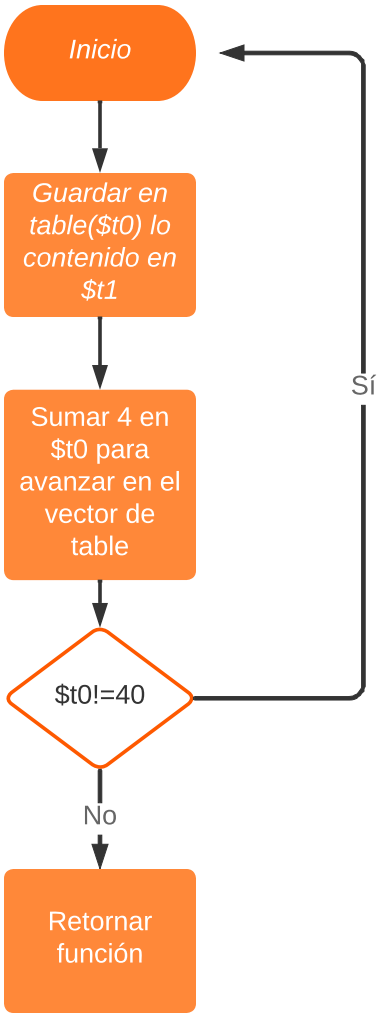
Llamar a la función drawMovement, encargada de dibujar la jugada en el tablero

Llamar a la función winner, encargada de verificar si hay ganador o un empate, imprime al ganador y verifica si se termina el juego

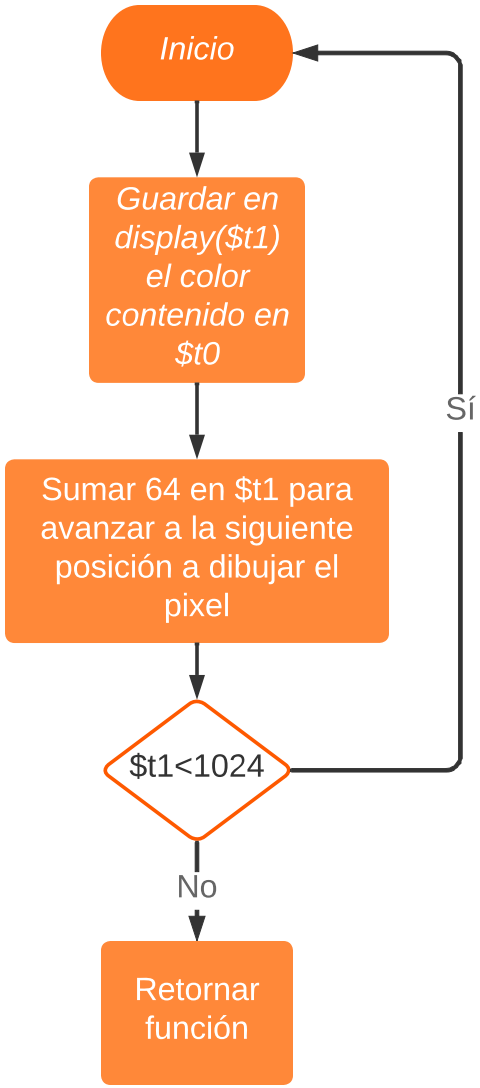
Llamar a la función changePlayer, encargada de cambiar al jugador

saltar a game

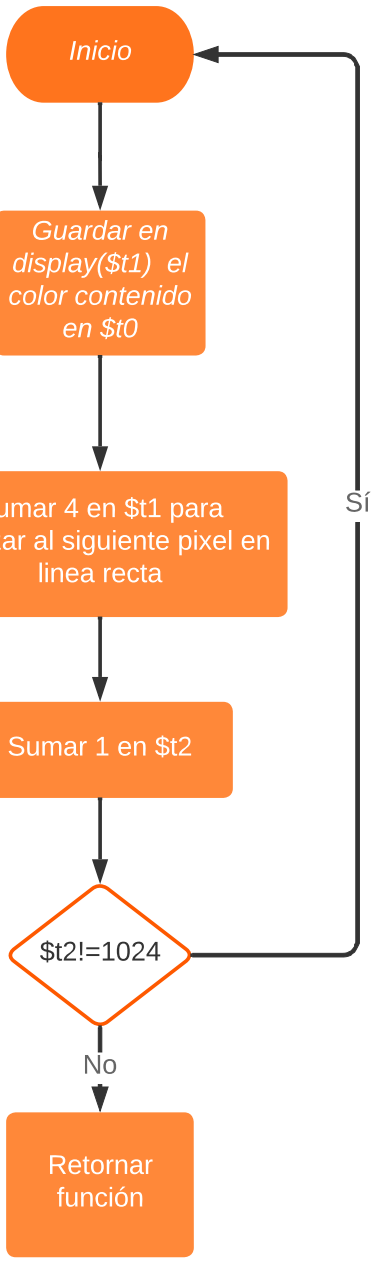
InitTable


Inicio
Guardar en table($t0) lo contenido en $t1
Sumar 4 en $t0 para avanzar en el vector de table
$t0!=40
Sí
No
Retornar función

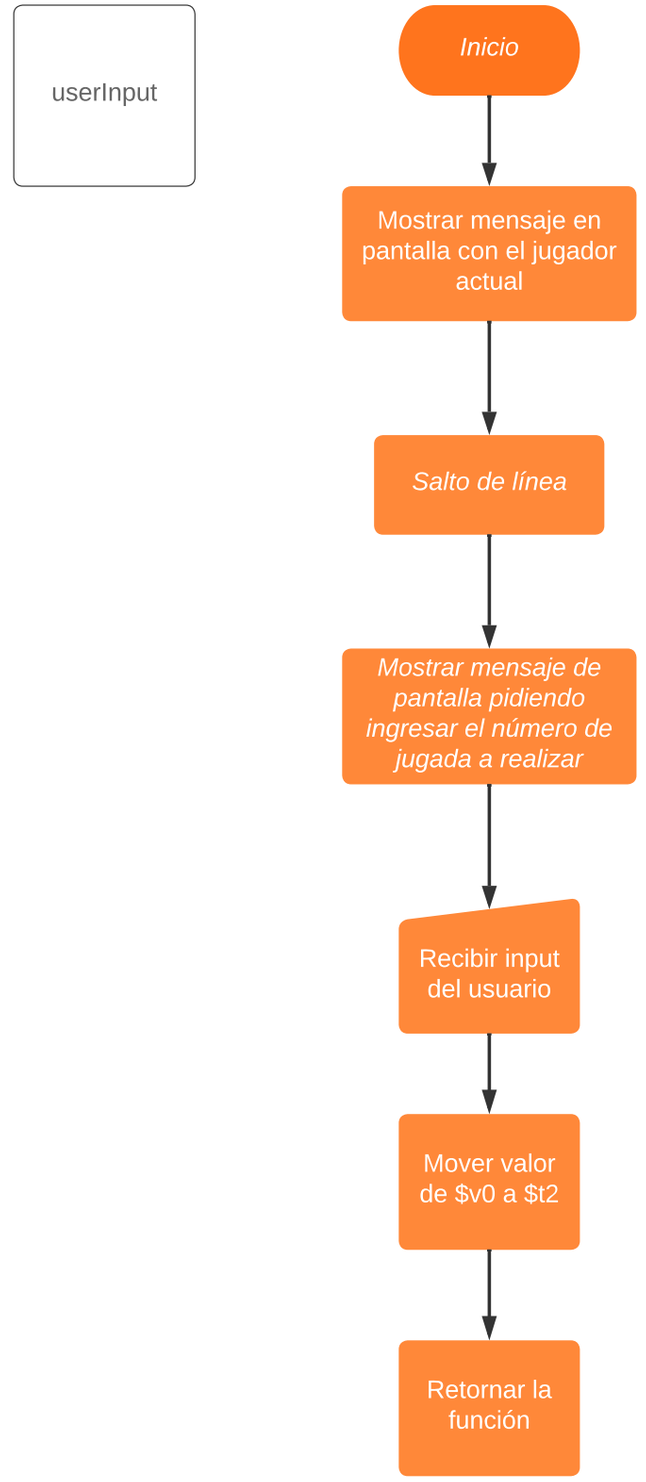
CreateVerticalLine



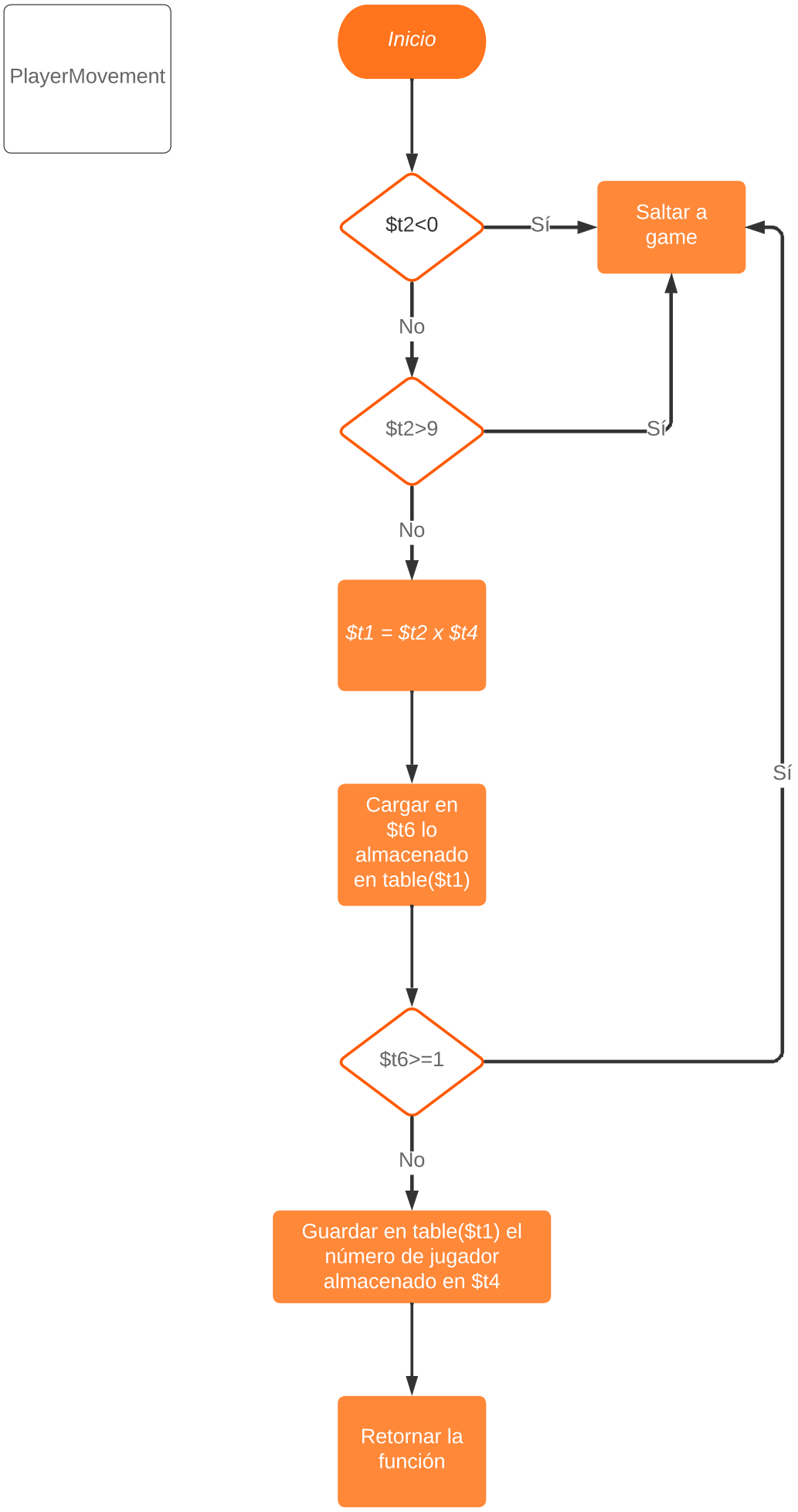
CreateHorizontalLine



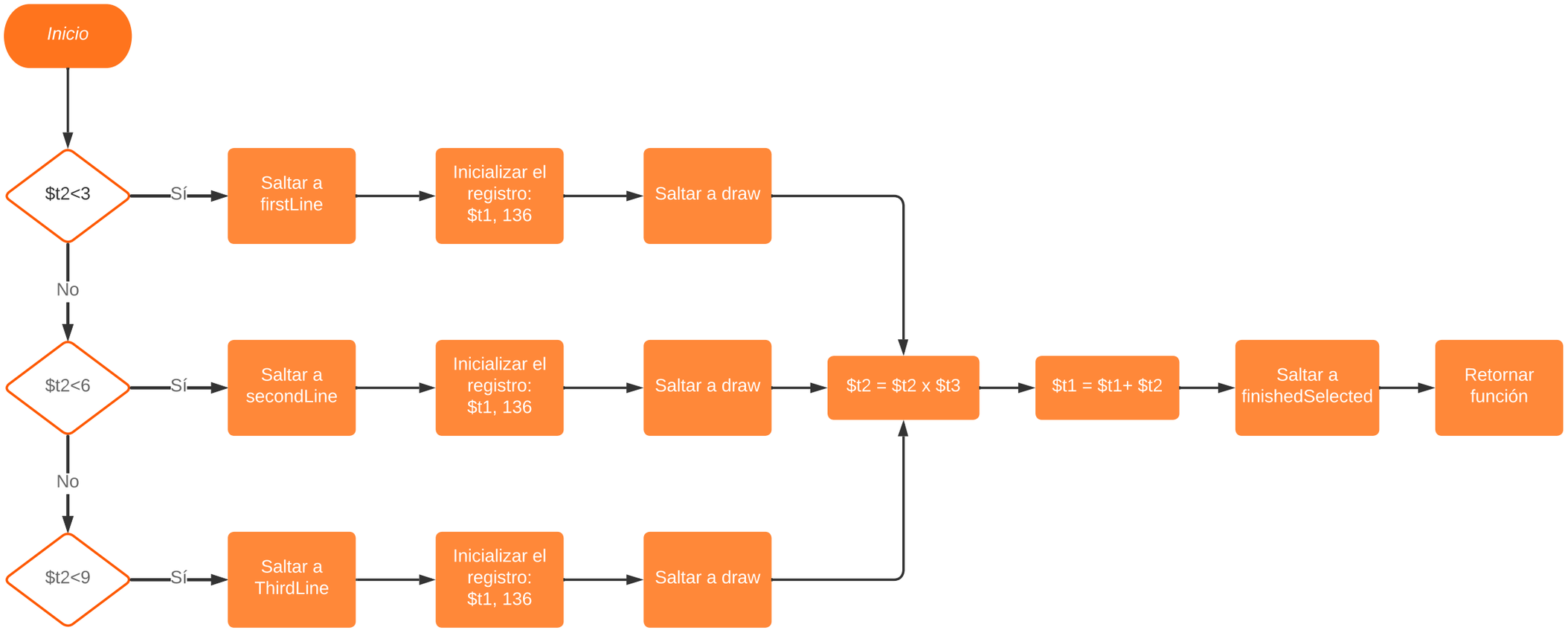
userInput

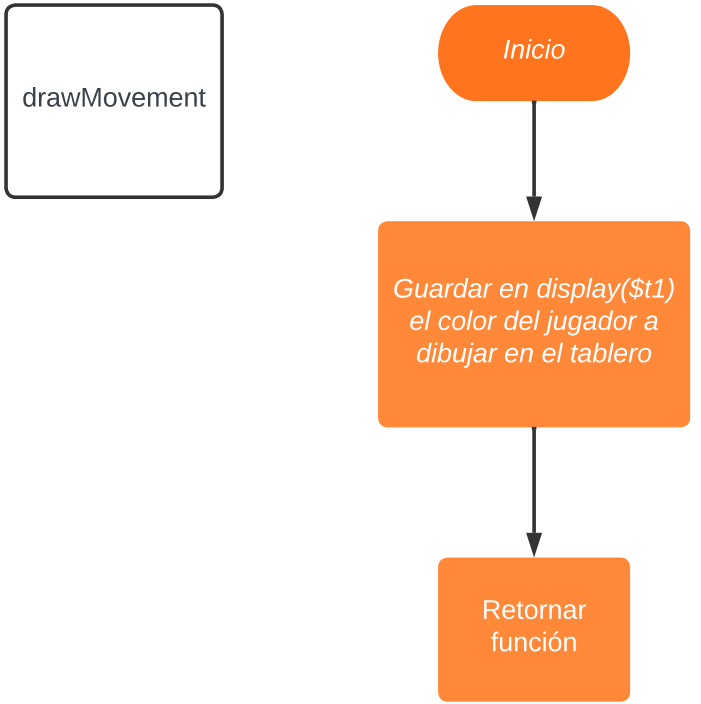

Inicio
Mostrar mensaje en pantalla con el jugador actual
Salto de línea
Mostrar mensaje de pantalla pidiendo ingresar el número de jugada a realizar
Recibir input del usuario
Mover valor de $v0 a $t2
Retornar la función

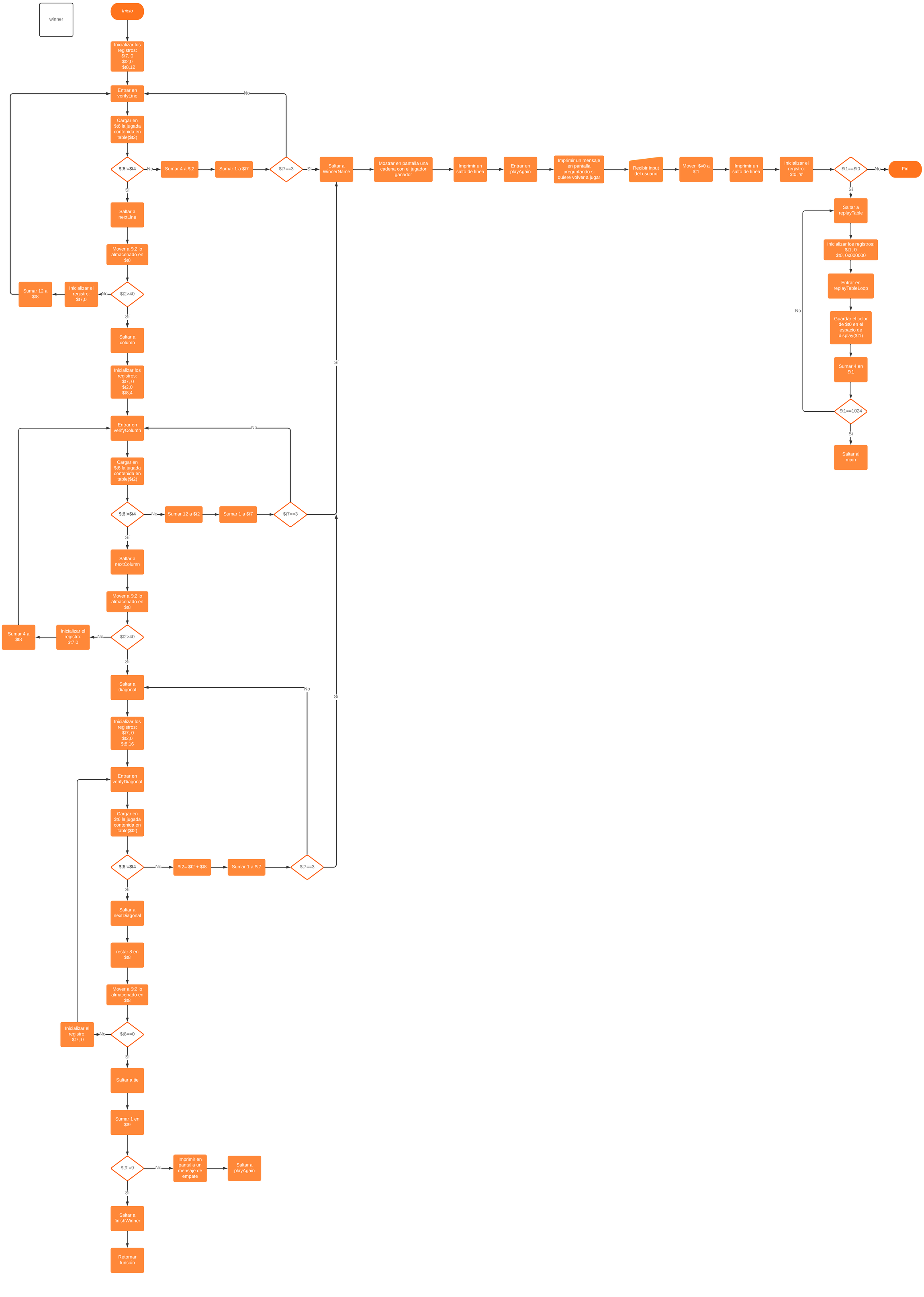
PlayerMovement



Selected


Inicio
$t2<3
Sí
Saltar a firstLine
Inicializar el registro: $t1, 136
Saltar a draw
No
$t2<6
Sí
Saltar a secondLine
Inicializar el registro: $t1, 136
Saltar a draw
No
$t2<9
Sí
Saltar a ThirdLine
Inicializar el registro: $t1, 136
Saltar a draw
$t2 = $t2 x $t3
$t1 = $t1+ $t2
Saltar a finishedSelected
Retornar función


drawMovement
Inicio
Guardar en display($t1) el color del jugador a dibujar en el tablero
Retornar función


winner
Inicio
Inicializar los registros: $t7, 0 $t2,0 $t8,12
Entrar en verifyLine
Cargar en $t6 la jugada contenida en table($t2)
$t6!=$t4
No
Sumar 4 a $t2
Sumar 1 a $t7
$t7==3
Sí
No
Saltar a WinnerName
Mostrar en pantalla una cadena con el jugador ganador
Imprimir un salto de línea
Entrar en playAgain
Imprimir un mensaje en pantalla preguntando si quiere volver a jugar
Recibir input del usuario
Mover $v0 a $t1
Imprimir un salto de línea
Inicializar el registro: $t0, 's'
$t1==$t0
No
Fin
Sí
Saltar a replayTable
Inicializar los registros: $t1, 0 $t0, 0x000000
Entrar en replayTableLoop
Guardar el color de $t0 en el espacio de display($t1)
Sumar 4 en $t1
$t1==1024
No
Sí
Saltar al main
Saltar a nextLine
Mover a $t2 lo almacenado en $t8
$t2>40
No
Inicializar el registro: $t7,0
Sumar 12 a $t8
Sí
Saltar a column
Inicializar los registros: $t7, 0 $t2,0 $t8,4
Entrar en verifyColumn
Cargar en $t6 la jugada contenida en table($t2)
$t6!=$t4
No
Sumar 12 a $t2
Sumar 1 a $t7
$t7==3
Sí
No
Saltar a nextColumn
Mover a $t2 lo almacenado en $t8
$t2>40
No
Inicializar el registro: $t7,0
Sumar 4 a $t8
Sí
Saltar a diagonal
Inicializar los registros: $t7, 0 $t2,0 $t8,16
Entrar en verifyDiagonal
Cargar en $t6 la jugada contenida en table($t2)
$t6!=$t4
No
$t2= $t2 + $t8
Sumar 1 a $t7
$t7==3
Sí
No
Saltar a nextDiagonal
restar 8 en $t8
Mover a $t2 lo almacenado en $t8
$t8==0
No
Inicializar el registro: $t7, 0
Sí
Saltar a tie
Sumar 1 en $t9
$t9!=9
No
Imprimir en pantalla un mensaje de empate
Saltar a playAgain
Sí
Saltar a finishWinner
Retornar función

changePlayer