<!--Estudio Shonos-->

Máquina Virtual

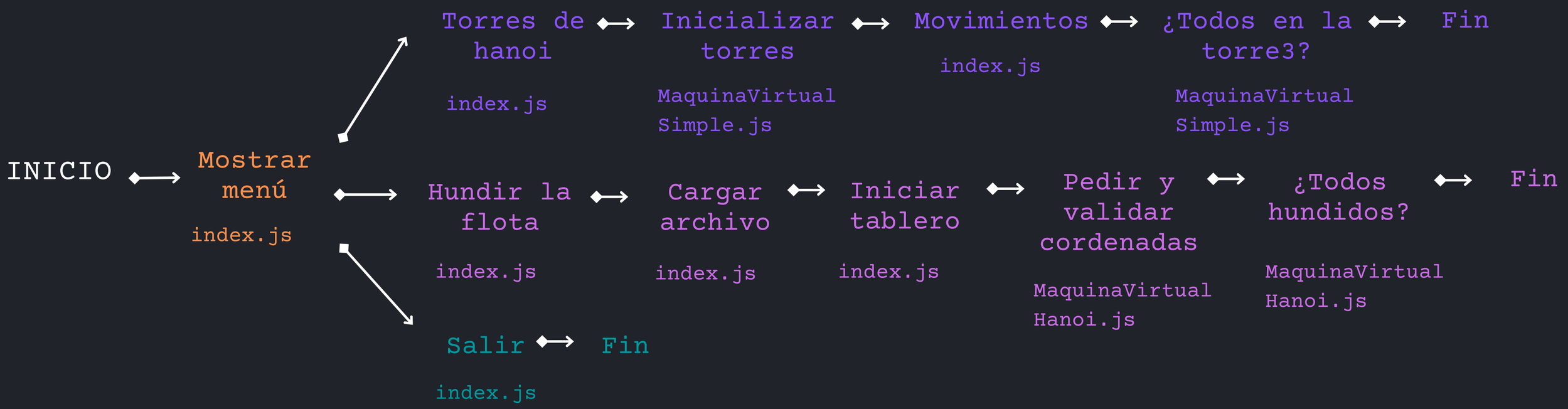
{

<Por="Marina Amaya y Cristina Barandiarán"/>

}

Contenidos

Diagrama del flujo {



}

Explicación del flujo {

Inicio del programa:

- readline -> gestionar las entradas.
- fs -> poder leer los archivos .mc
- MaquinaVirtualSimple.js -> gestionar la lógica del juego hundir la flota)
- MaquinaVirtualHanoi.js -> gestionar la lógica del juego torres de hanoi)

Inicialización:

Se realiza la interfaz y un conjunto para ratrear las coordenadas de los disparos y también creamos un tablero 10x10 lleno de ~ azules que representan el agua.

}

Menú Principal {

Función mostrarMenu():

- **Muestra las opciones de juego al usuario:**
  - **1. Torres de Hanoi**
  - **2. Hundir la flota**
  - **3. Salir**

Entrada del Usuario para Seleccionar Opción:

- **Se pregunta al usuario qué juego quiere jugar.**
- **Según la opción ingresada, se llama a la función correspondiente:**
  - **1: Llama a jugarTorresDeHanoi().**
  - **2: Llama a iniciarHundirLaFlota().**
  - **3: Termina el programa con un mensaje de despedida.**

```
// Función para mostrar el menú principal
function mostrarMenu() {
    console.log("Elige un juego:"); // Mostrar las opciones del menú
    console.log("1. Torres de Hanoi");
    console.log("2. Hundir la flota");
    console.log("3. Salir");
}
```

}

Flujo del Juego: Torres de Hanoi {

1. Se pregunta al usuario cuántos discos quiere usar.
2. Inicialización del Juego:
   - La clase MaquinaVirtualHanoi inicializa tres pilas que representan las tres torres.
   - Colocando todos los discos en la primera torre.
3. Se pide al usuario que ingrese un movimiento en el formato "deTorre aTorre" (por ejemplo, "1,3").
   - Se validan las entradas y si es valido se actualiza el estado.
   - La función mover() de MaquinaVirtualHanoi realiza el movimiento y devuelve si fue exitoso.
4. Si el usuario ha movido todos los discos a la última torre en el orden correcto, se muestra un mensaje de victoria.

```
// Función para jugar Torres de Hanoi
function jugarTorresDeHanoi() {
    const juegoHanoi = new TorresDeHanoi(); // Instanciar el juego Torres de Hanoi
    rl.question("¿Cuántos discos quieres usar? ", (numDiscos) => { // Preguntar cuántos
        juegoHanoi.inicializar(parseInt(numDiscos)); // Inicializar el juego con el núm

        // Función para mostrar el estado actual de las torres
        const mostrarEstadoTorres = () => {
            console.log("Estado actual de las torres:");
            console.log(`Torre 1: ${juegoHanoi.torres[0].items.join(' ')}`); // Mostrar
            console.log(`Torre 2: ${juegoHanoi.torres[1].items.join(' ')}`); // Mostrar
            console.log(`Torre 3: ${juegoHanoi.torres[2].items.join(' ')}\n`); // Mostr
        };

        mostrarEstadoTorres(); // Llamar a la función para mostrar el estado inicial
    // Función para solicitar el movimiento del usuario
    const solicitarMovimiento = () => {
        rl.question("Ingresa el movimiento (deTorre aTorre) o 'exit' para salir: ", (input) => { // Preguntar
            if (input.toLowerCase() === 'exit') { // Verificar si el usuario desea salir
                console.log("Saliendo del juego de Torres de Hanoi.");
                rl.close(); // Cerrar la interfaz de readline
                return; // Salir de la función
            }

            const [deTorre, aTorre] = input.split(',').map(Number); // Parsear el movimiento ingresado

            // Verificar si las torres ingresadas son válidas
            if (isNaN(deTorre) || isNaN(aTorre) || deTorre < 1 || deTorre > 3 || aTorre < 1 || aTorre > 3) {
                console.log("Entrada no válida. Debes ingresar números del 1 al 3.");
                return solicitarMovimiento(); // Solicitar nuevamente el movimiento
            }

            const movimientoExitoso = juegoHanoi.mover(deTorre - 1, aTorre - 1); // Intentar mover el disco
            if (movimientoExitoso) {
                console.log(`Moviendo disco de la Torre ${deTorre} a la Torre ${aTorre}.`);
                mostrarEstadoTorres(); // Mostrar el estado actualizado de las torres
            } else {
                console.log("Movimiento no permitido."); // Informar de un movimiento no válido
            }

            if (juegoHanoi.haGanado()) { // Verificar si el jugador ha ganado
                console.log("¡Felicidades, has ganado!");
                rl.close(); // Cerrar la interfaz de readline
            } else {
                solicitarMovimiento(); // Solicitar otro movimiento
            }
        });
    };
}
```

Flujo del Juego: Hundir la Flota{

1. Listar Archivos .mc:
- Lee el contenido del directorio y filtra los archivos que terminan en .mc, si encuentra archivos, muestra una lista de los archivos .mc disponibles y se pide al usuario que seleccione uno para cargarlo.

2. Se carga el archivo seleccionado y llama a un método para procesar las instrucciones del archivo.
3. Mostramos el estado inicial.
4. Realizar Disparos.
5. El juego acaba cuando:

o Hundes todos los barcos mediante disparos
o Introduciendo exit.

```
function iniciarHundirLaFlota() {
    console.log("Iniciando Hundir la flota..."); // Mensaje inicial del juego

    listarArchivosMC((archivos) => { // Llamar a la función para listar archivos
        if (archivos.length === 0) { // Verificar si no hay archivos .mc
            console.log("No se encontraron archivos .mc. Asegúrate de que haya archivos en el directorio.");
            rl.close(); // Cerrar la interfaz de readline
            return;
        }

        console.log("Selecciona un archivo .mc:");
        archivos.forEach((archivo, index) => { // Listar los archivos disponibles
            console.log(`${index + 1}. ${archivo}`);
        });
            // Función para realizar un disparo en Hundir la flota
            const disparar = () => {
                maquina.imprimirTablero(); // Mostrar el tablero

                rl.question('Ingresa las coordenadas para disparar (x,y) o "exit" para salir: ', (input) => { // Pedir las coordenadas
                    if (input.toLowerCase() === 'exit') { // Verificar si el usuario desea salir
                        console.log("Saliendo del juego de Hundir la flota.");
                        rl.close(); // Cerrar la interfaz de readline
                        return;
                    }

                    const [x, y] = input.split(',').map(Number); // Parsear las coordenadas

                    // Verificar si las coordenadas están dentro del rango permitido
                    if (x < 0 || x >= 10 || y < 0 || y >= 10) {
                        console.log("Coordenadas fuera del rango. Debes ingresar valores entre 0 y 9.");
                        return disparar(); // Volver a solicitar coordenadas
                    }

                    if (!disparosRealizados.has(`${x},${y}`)) { // Verificar si ya se ha disparado en esas coordenadas
                        disparosRealizados.add(`${x},${y}`); // Agregar las coordenadas a la lista de disparos realizados
                        const resultado = maquina.disparar(x, y); // Realizar el disparo
                        if (resultado) {
                            console.log(`Disparo en (${x}, ${y}) exitoso.`);
                        } else {
                            console.log(`Fallaste en (${x}, ${y}).`);
                        }

                        if (maquina.todosLosBarcosHundidos()) { // Verificar si el jugador ha hundido todos los barcos
                            console.log("¡Has hundido todos los barcos! Fin del juego.");
                            rl.close(); // Cerrar la interfaz de readline
                        } else {
                            disparar(); // Volver a solicitar un disparo
                        }
                    } else {
                        console.log("Ya has disparado en esas coordenadas. Intenta con otras.");
                        disparar(); // Volver a solicitar un disparo
                    }
                });
            };

            disparar(); // Iniciar la secuencia de disparos
        });
    });
}
```

}

Uso de las pilas en el programa {

Utilizamos pilas como estructuras de datos para gestionar los elementos del juego. En la clase MaquinaVirtualSimple, se emplea una pila (pilaBarcos) para almacenar los barcos creados y una pila temporal para manejar los barcos durante los disparos, asegurando que se respeten las reglas del juego. Por otro lado, en la clase MaquinaVirtualHanoi, se utilizan tres pilas, cada una representando una torre, para apilar discos y permitir movimientos válidos de acuerdo con las reglas del juego. En ambos casos, las pilas facilitan la manipulación de los elementos del juego y el seguimiento del estado del mismo de manera lógica y organizada.

```
Elige un juego:
1. Torres de Hanoi
2. Hundir la flota
3. Salir
Selecciona una opción: 1
¿Cuántos discos quieres usar? 3
Estado actual de las torres:
Torre 1: 3 2 1
Torre 2:
Torre 3:

Ingresa el movimiento (deTorre aTorre) o 'exit' para salir:
```

```
Elige un juego:
1. Torres de Hanoi
2. Hundir la flota
3. Salir
Selecciona una opción: 2
Iniciando Hundir la flota...
Selecciona un archivo .mc:
1. configuracion1.mc
2. configuracion2.mc
3. configuracion3.mc
Ingresa el número del archivo que deseas cargar: 1
Tablero:
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
Ingresa las coordenadas para disparar (x,y) o "exit" para salir: 1,1
Acierto en (1, 1) al barco barco_1_1!
Disparo en (1, 1) exitoso.
Tablero:
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ X ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
~ ~ ~ ~ ~ ~ ~ ~ ~ ~
Ingresa las coordenadas para disparar (x,y) o "exit" para salir:
```

}

<!--Estudio Shonos-->

Gracias {

<Por="Marina Amaya y Cristina Barandiarán"/>

}