MARCO TEÓRICO.

Batalla naval es un juego tradicional de estrategia y algo de suerte, que normalmente involucra a dos participantes. Sin embargo, esta versión del juego, desarrollado en MatLab, será de un solo participante contra la máquina. Tanto el jugador como la máquina manejan dos tableros o matrices: un *tablero de tiro* dividido en casillas en donde va ingresando las coordenadas de donde quiere disparar a tres barcos por medio de la ventana de comandos de MatLab, y un *tablero de océano o mapa* donde están puestas las tres naves.

Cuando se inicia el juego, el jugador indicará la dificultad, la cual estará dada por la cantidad de casillas que tengan los cuatro tableros, que son las dimensiones 8x8, 9x9 y 10x10. Los tres barcos tienen longitudes diferentes (3, 4 y 5) y los dos mapas se configuran aleatoriamente antes de comenzar el juego. Sólo será necesario que se muestren en pantalla el mapa y el tablero de tiro del jugador.

Cuando comienza el desarrollo del juego, tanto el jugador y la máquina irán disparando por turnos al mapa contrario a través de su tablero de tiro (la máquina, aleatoriamente y el jugador, a través de la ventana de comandos). El mapa del jugador diferenciará los impactos y tiros fallidos que hace la máquina contra él con colores distintos. Lo mismo sucede con el tablero de tiro del jugador, diferenciando sus impactos y sus fallos contra la máquina. Un barco se hunde si recibió un golpe mínimo.

El juego termina cuando la máquina o el jugador hundan los tres barcos del contrincante o el jugador decida rendirse.

Se establecerán las reglas del juego para poder diseñarlo, crear pseudocódigo o diagrama de flujo y finalmente crear el código.

Reglas:

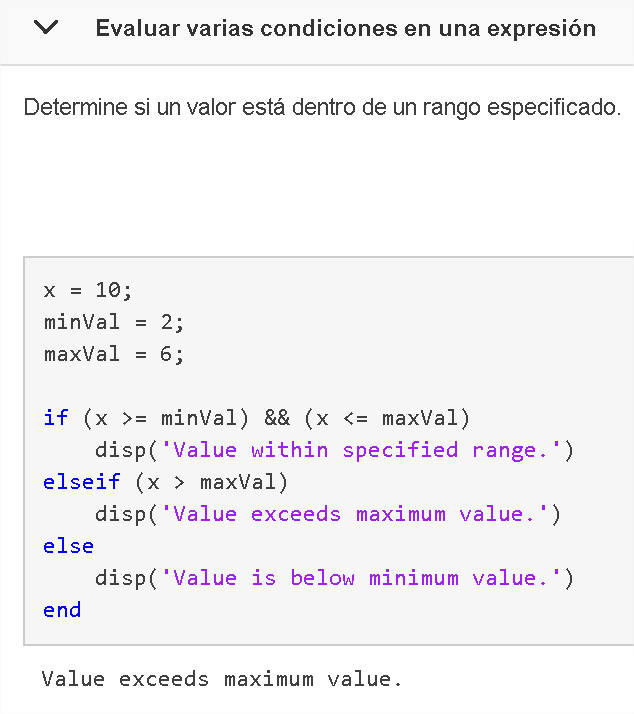
* 1. Un jugador debe elegir la dificultad y los 3 barcos de los 2 mapas se generan aleatoriamente en un mapa o cuadrícula.
  2. Los barcos tienen longitudes de 3 (lancha), 4 (acorazado) y 5 (porta-aviones).
  3. Debe elegir la orientación de cada barco. Sólo es posible el sentido vertical y el horizontal. No se permite el sentido diagonal.
  4. No puede solapar barcos, ni salirse de la matriz.
  5. Debe aniquilar a todas las naves; es decir, que todas las naves enemigas hayan recibido un impacto.
  6. El participante puede retirarse cuando lo desee.
  7. Al terminar el juego, se muestra un mensaje en pantalla de finalización con la cantidad de aciertos y disparos totales

Sentencias de control de flujo.

Se necesitará utilizar condicionales y ciclos para realizar validaciones (preguntar si se ingresaron bien los datos) y hacer acciones repetitivas como llenar la matriz de barquitos o disparar al tablero.

1. Condicionales.
   1. Sentencia IF - ELSE IF - ELSE.

*Si* una condición dada es verdadera (comparación), se ejecutarán unas órdenes una sola vez dentro del bloque IF. *Si no* se cumplen, *si* otra condición opcional es verdadera, ejecuta el conjunto de órdenes dentro de ELSE IF. *Si ninguna* *de las anteriores* es verdadera, opcional puede ejecutarse otro bloque de código.

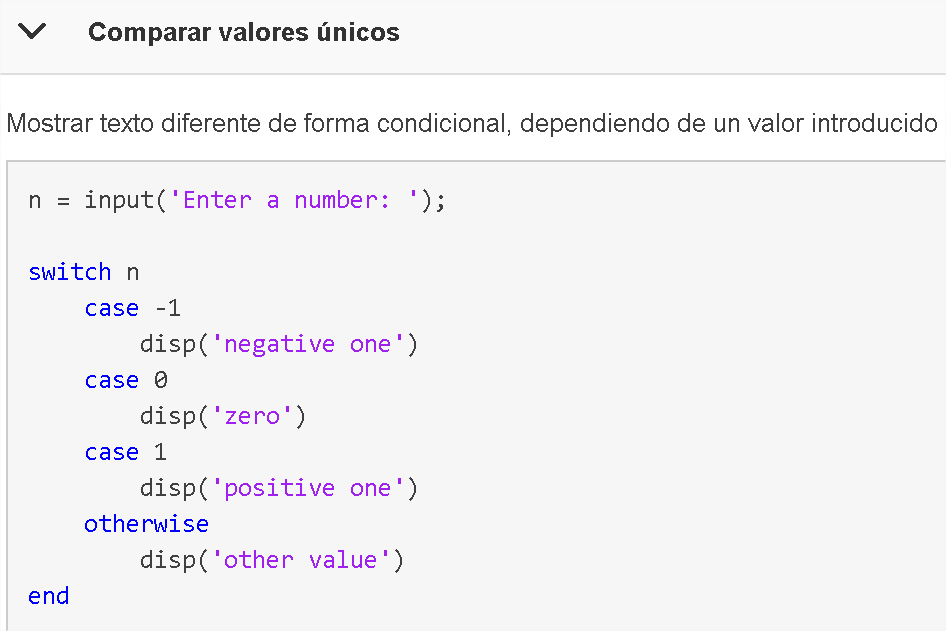


Recuperado de la documentación oficial de Mathworks.

Aplicación en el juego: IF servirá para verificar si se solapan los barcos y si se salen de la matriz.

* 1. Sentencia SWITCH – CASE - OTHERWISE

Se usa en la decisión de ir por un camino entre varias opciones de acuerdo al *caso* en que una variable tenga un valor, o si la variable no tiene ningún valor entre los supuestos, puede ejecutarse otras opciones *de cualquier forma*.

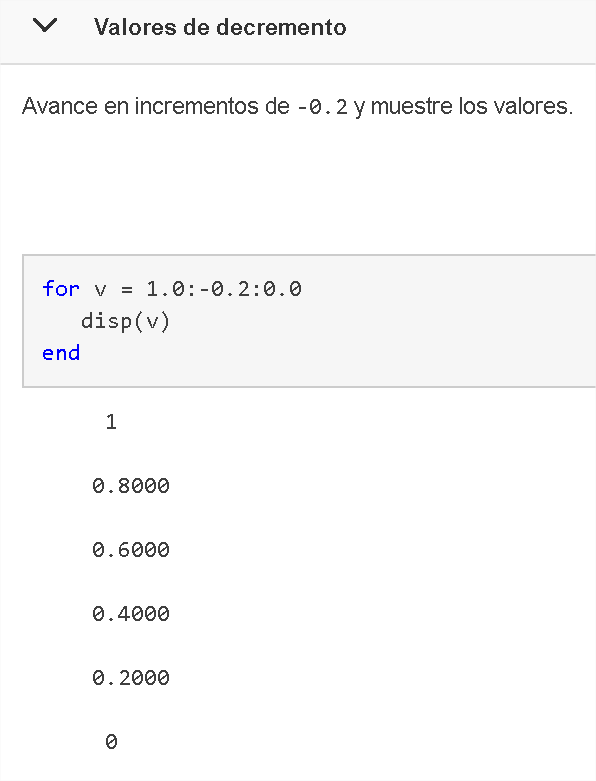


Recuperado de la documentación oficial de Mathworks

Si se introduce un 2, se imprime en pantalla *‘other value’*.

1. Ciclos.
   1. Bucle FOR.

*Para* una variable que tomará valores de un vector que tiene un valor inicial, uno final y un tamaño de paso, se realizará un conjunto de acciones repetitivas una determinada cantidad de veces cada que dicha variable adquiera el valor siguiente dentro del vector.

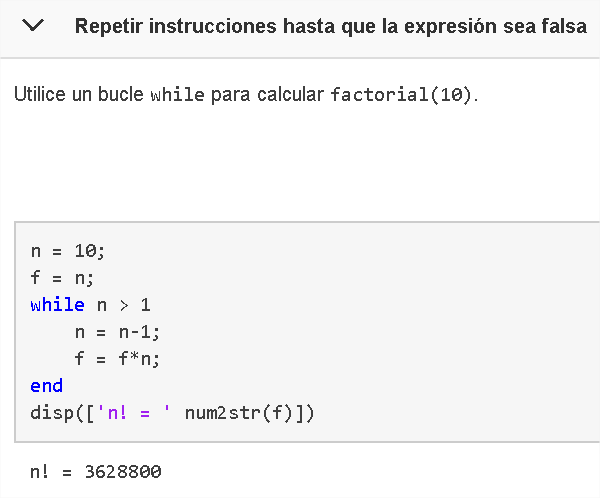


Recuperado de la documentación oficial de Mathworks.

Aplicación en el juego: el ciclo FOR se puede usar para asignar los números en la matriz según la primera imagen según el barco ingresado.

* 1. Bucle WHILE.

Se utiliza para realizar unas órdenes repetitivas indefinidamente *siempre que* una condición evaluada sea verdadera (tipo IF).



Recuperado de la documentación oficial de Mathworks

Aplicación en el juego: este ciclo se utilizará para el desarrollo del juego donde el jugador está disparando al azar al tablero.

El juego termina cuando este ciclo termina, pues la condición del WHILE es si la matriz tiene barcos sin hundir.

1. Lógica de vectores.
   1. Indexación.

Matriz(x,y): se accede al elemento de la fila x, columna y.

* 1. Asignación.

Matriz(x,y) = a: el elemento de la fila x, columna y, adquiere el valor de a. Útil para sobrescribir un valor dentro de la matriz al hacer un disparo.

* 1. Comparación.

Matriz(x,y) == 0: verdadero si el elemento de la matriz es igual a cero. Útil para preguntar por barcos disponibles para disparar.

**BIBLIOGRAFIA:**

* <https://la.mathworks.com/help/matlab/learn_matlab/matrices-and-arrays.html>
* <https://la.mathworks.com/help/matlab/ref/if.html>
* <https://la.mathworks.com/help/matlab/ref/switch.html>
* <https://tutorias.co/arrays-matlab-ubicacion-barco-batalla-naval/>