

HUNTER GRID

1. Descripción general

Hunter Grid es un juego donde el jugador debe atrapar a un “super-barco” llamado **Hunter**, que se mueve por el grid a gran velocidad.

El objetivo es **clicar exactamente la celda donde está el Hunter**, antes de quedarse sin vidas.

Además, el grid contiene **trampas invisibles**. Al clicar en una trampa:

- El jugador pierde 1 vida.
- La trampa se revela permanentemente.

Cada vez que el jugador acierta al Hunter:

- Las trampas se **vuelven a generar aleatoriamente**, evitando la celda del Hunter.

El jugador **gana** al atrapar al Hunter 10 veces.

El jugador **pierde** al llegar a 0 vidas.

2. Formulario inicial

Antes de empezar el juego, el usuario debe introducir:

1. Número de filas (≥ 3 y ≤ 10)
2. Número de columnas (≥ 3 y ≤ 10)
3. Velocidad del Hunter (entre 200 ms y 800 ms)
4. Número de trampas (mínimo 1, máximo 1/3 de las celdas)

Validación obligatoria con **HTML5 + JS** (API de Validación).

Una vez validado, se genera el grid.

3. Generación del Grid

El grid debe respetar:

- Tamaño exacto según el formulario.
- Cada celda debe ser **clicable**.
- Las trampas se colocan aleatoriamente.
- El Hunter comienza en una **celda aleatoria que no sea una trampa**.
- Las trampas **no deben mostrarse** al jugador, tampoco el hunter.

4. Movimiento del Hunter

El Hunter se mueve de celda en celda automáticamente:

- Movimiento cada **X ms**, donde X es la velocidad del formulario.
- La nueva celda debe cumplirse:
 - No puede ser la celda actual.
 - **No puede ser una celda con trampa.**
- Se recomienda usar `setInterval()`.

5. Interacción del jugador

El jugador puede clicar cualquier celda del grid:

Si clicas la celda del Hunter:

- Ganas +1 captura.
- Se muestra clara visualización del acierto, por ejemplo, un alert o mensaje.
- La velocidad de movimiento del barco aumenta.
- **Las trampas se regeneran completamente** en nuevas posiciones aleatorias.

Si clicas una trampa:

- Pierdes 1 vida.
- La celda muestra una **X** permanente, hasta la siguiente partida.
- Si quedas en 0 vidas → derrota.

6. Condiciones de victoria y derrota

Victoria

El jugador gana cuando ha atrapado al Hunter **10 veces**.

Se debe mostrar un popup:

- "¡Has ganado!"
- Al aceptar reinicia la partida.

Derrota

Ocurre cuando: Las vidas llegan a 0. Aparece: "Has perdido" y Botón **Volver a jugar**

7. Persistencia con LocalStorage

Debe guardarse:

- Configuración del formulario (filas, columnas, trampas, velocidad)
- Número total de capturas logradas
- Mejor tiempo registrado
- Fecha de la última partida

Al recargar la página:

- Se restaura **solo la configuración del formulario**.
- NO se restauran trampas ni la posición del Hunter (se regeneran).
- Las vidas vuelven a su máximo (normalmente 3).

8. Interfaz obligatoria

La interfaz debe contener:

- Formulario inicial
- Panel con:
 - Vidas restantes
 - Capturas realizadas / 10
 - Temporizador de partida
- Grid dinámico
- Popups de victoria/derrota

9. Funciones recomendadas (JS)

- `initGame()`
- `generateGrid()`
- `placeHunter()`
- `placeTraps()`
- `moveHunter()`
- `onCellClick(event)`
- `resetTrapsOnCapture()`
- `checkWin()`

- checkLose()
- saveStateToLocalStorage()
- loadFormFromLocalStorage()

10. Rúbrica de evaluación

Criterio	Puntos
Generación del grid según formulario	2
Movimiento del Hunter (aleatorio + válido)	2
Lógica de clic (Hunter, trampas, vidas)	2
Regeneración aleatoria de trampas tras acierto	2
LocalStorage + popup final	1,5
HTML5/PATRONES	0,5
Total	10 puntos