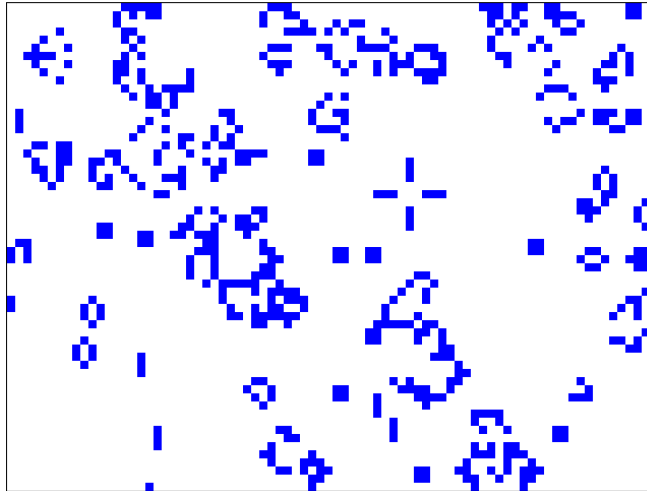


## AEV\_2 Juego de la vida



### Ejercicio 1: Cambiar el tamaño y color de las células

- Modifica las constantes `CELL_SIZE` y el color de las células vivas (`ctx.fillStyle`) para que las celdas sean de **12 píxeles** y el color sea **verde lima** ("limegreen").
- Prueba cómo cambia la apariencia del tablero.

---

### Ejercicio 2: Implementar pausa y reanudación con la barra espaciadora

- Descomenta el bloque del evento `keydown` y asegúrate de que la simulación se detenga y continúe al presionar la tecla **espacio**.
- Agrega un mensaje en consola que diga "Simulación pausada" o "Simulación en ejecución" según el estado.

---

### Ejercicio 3: Añadir un botón para reiniciar la simulación

- Crea un botón en el HTML llamado **"Reiniciar"** (`<button id="resetBtn">Reiniciar</button>`).
- Asigna un evento `onclick` que vacíe el grid (todas las células muertas) y vuelva a llamar a `randomize(0.2)` para crear una nueva población aleatoria.

---

### Ejercicio 4: Mostrar el número de generaciones

- Crea una variable `let generations = 0;`
- En cada llamada de `step()`, incrementa el contador y muestra el número de generación en pantalla (por ejemplo, en un elemento `<p id="info">Generación: X</p>`).



💡 *Objetivo:* practicar actualizaciones dinámicas del DOM a partir del bucle de animación.

---

### Ejercicio 5: Genera un patrones personalizados

Implementa los patrones clásicos del juego de la vida: **Blinker**, **Toad**, **Beacon** y **Glider** en un canvas vacío. Puedes añadir tantos como quieras en las coordenadas que propongas.

💡 *Objetivo:* comprender y reproducir los patrones osciladores y móviles clásicos del Juego de la Vida.

---

### ⚙️ Ejercicio 6 (Avanzado): Análisis de rendimiento con Lighthouse y Performance

#### Parte 1 — Reporte de Lighthouse

1. Abre la página del proyecto en **Google Chrome**.
  2. Abre las **DevTools** (Ctrl+Shift+I o Cmd+Option+I).
  3. Ve a la pestaña “**Lighthouse**”.
  4. Genera un **reporte de rendimiento** (“**Performance**”).
  5. Guarda el resultado en formato HTML o PDF y analiza:
    - **Performance Score**
    - **Tiempo hasta el primer renderizado (First Contentful Paint)**
    - **Tiempo hasta la interacción (TTI)**
- 

#### Parte 2 — Análisis en la pestaña “Performance”

1. Abre la pestaña **Performance** en las DevTools.
2. Inicia la grabación y deja correr el juego por unos segundos.
3. Detén la grabación y examina los resultados.
4. Identifica:
  - Qué función o bloque de código consume **más tiempo de CPU** (por ejemplo, `draw()`, `step()`, o `neighbors()`).
  - Si hay **repaints** o **reflows** excesivos.



**Entrega:** La entrega se realizará via floridaOberta creando un pequeño video explicativo del código desarrollado (5-10 min) en el que se expliquen los scripts creados y las dificultades/mejoras posibles,

Carpeta comprimida con todos los scripts en floridaOberta con el formato “Apellido1Apellido2Nombre\_AeV2.zip”, opcionalmente se pueden subir los ejercicios a Github y añadir un enlace al repositorio.