<!DOCTYPE html>

<html lang="fr">

<head>

  <meta charset="utf-8" />

  <meta name="viewport" content="width=device-width, initial-scale=1" />

  <title>War-ixel</title>

  <style>

    :root{

      --deep:#255c9e;

      --shallow:#8fd0ff;

      --ai:#b71c1c;

      --select:#ffcc00;

      --reach-fill:  rgba(60,170,120,.32);

      --theo-fill:   rgba(100,140,220,.20);

      --bg:#eae4d6;

      --ink:#171717;

      --paper1:#f4e2b8; --paper2:#f0d59a; --paper3:#e8c77a; --paper-ink:#473924;

    }

    \*{box-sizing:border-box}

    body{margin:0;background:var(--bg);color:var(--ink);font-family:system-ui,-apple-system,Segoe UI,Roboto,Arial,sans-serif}

    /\* Header \*/

    header{

      position:sticky; top:0; z-index:20;

      padding:10px 14px;

      display:flex;flex-wrap:wrap;gap:8px;align-items:center;justify-content:flex-start;

      color:var(--paper-ink);

      border-bottom:1px solid #c8b99c;

      background:

        linear-gradient(180deg,#fff7 0 10%, #0000 25% 75%, #fff6 90% 100%),

        radial-gradient(55% 120% at -5% -40%, #0002 0 40%, #0000 60%),

        radial-gradient(45% 110% at 110% -40%, #0002 0 40%, #0000 60%),

        linear-gradient(180deg,var(--paper1),var(--paper2) 55%,var(--paper3));

      box-shadow:inset 0 1px 0 #fff7,inset 0 -1px 0 #0001,0 4px 16px rgba(0,0,0,.12);

    }

    .paper-chip{padding:6px 10px;border-radius:9px;border:1px solid #d7c7ad;background:#fff9;backdrop-filter:blur(2px)}

    .controls{display:flex;gap:8px;align-items:center}

    .controls button{padding:8px 12px;border:1px solid #cdbb98;border-radius:9px;background:#fff9;cursor:pointer;box-shadow:inset 0 1px 0 #fff7}

    /\* Layout \*/

    .layout{display:grid;grid-template-columns:320px 1fr;gap:12px;padding:12px}

    #sidebar{display:flex;flex-direction:column;gap:12px}

    .paper{

      border-radius:14px;border:1px solid #d7c7ad;color:var(--paper-ink);padding:12px 14px;

      background:

        linear-gradient(180deg,#fff9 0 10%, #0000 30% 70%, #fff6 90% 100%),

        repeating-linear-gradient( 6deg, #0000 0 12px, #0001 12px 13px),

        linear-gradient(180deg, var(--paper1), var(--paper2) 55%, var(--paper3));

      box-shadow:inset 0 1px 0 #fff7,inset 0 -1px 0 #0001,0 10px 24px rgba(0,0,0,.18);

    }

    .paper h3{margin:0 0 8px 0;color:#3b2f20}

    #actions{display:grid;gap:6px}

    #actions button{padding:8px;border-radius:8px;border:1px solid #cdbb98;background:#fff;cursor:pointer;text-align:left}

    #actions button[disabled]{opacity:.45;cursor:not-allowed}

    /\* Journal \*/

    #log{max-height:240px;overflow:auto}

    .entry{border-bottom:1px dashed #d7c7ad;padding:6px 0}

    .entry:last-child{border-bottom:0}

    .summary{cursor:pointer;font-weight:700}

    .detail{max-height:0;overflow:hidden;transition:max-height .25s ease;font-size:12px;color:#3f3527}

    .entry.open .detail{max-height:260px}

    .entry[data-owner="1"] .summary{color:var(--my)}

    .entry[data-owner="2"] .summary{color:var(--ai)}

    .entry[data-owner="3"] .summary{color:#6a1b9a}

    .entry[data-owner="4"] .summary{color:#e39a1d}

    .entry[data-owner="5"] .summary{color:#00897b}

    /\* Board \*/

    #board{

      position:relative;border:1px solid #c8bfae;border-radius:12px;padding:18px;overflow:auto;

      display:flex;align-items:center;justify-content:center;

      background:

        repeating-linear-gradient(45deg, #cfe7ff 0 18px, #d8edff 18px 36px),

        radial-gradient(80% 60% at 0% 10%, #0000000b 0 50%, #0000 70%),

        radial-gradient(70% 70% at 100% 90%, #0000000d 0 50%, #0000 75%),

        linear-gradient(180deg, #fff, #fdfbf7 55%, #faf6ef);

    }

    #stage{position:relative;margin:auto}

    #terrain, #overlay, #grid{position:absolute;left:0;top:0}

    #overlay{pointer-events:none;z-index:2}

    #grid{z-index:3}

    .hex{position:absolute;left:0;top:0;transform:translate(-50%,-50%);cursor:pointer}

    .hex.selected{outline:3px solid var(--select);outline-offset:-2px}

    .badge{

      position:absolute;left:50%;top:50%;transform:translate(-50%,-50%);

      width:34px;height:34px;border-radius:50%;

      display:flex;align-items:center;justify-content:center;

      border:2px solid rgba(255,255,255,.85);box-shadow:0 1px 1px rgba(0,0,0,.08)

    }

    .badge .icon{width:18px;height:18px;display:block}

    .badge::after{content:none !important}

    .camo{opacity:.82}

    .camo::before{content:"";position:absolute;inset:-2px;background:repeating-linear-gradient(135deg,transparent 0 6px,rgba(0,0,0,.06) 6px 12px);mix-blend-mode:multiply;border-radius:8px}

    .tower,.city{position:absolute;left:50%;top:50%;transform:translate(-50%,-50%);width:32px;height:28px;border-radius:6px;border:2px solid rgba(0,0,0,.15)}

    .city{background:#c68c53;display:grid;place-items:center}

    .tower{background:#e0e0e0;display:grid;place-items:center}

    .tower svg{width:22px;height:22px}

    .castle{width:22px;height:16px}

    .ship{

      width:42px;height:26px;border-radius:6px;color:#fff;border:2px solid rgba(255,255,255,.85);

      display:flex;align-items:center;justify-content:center;position:absolute;left:50%;top:50%;transform:translate(-50%,-50%);

      box-shadow:0 1px 1px rgba(0,0,0,.08)

    }

    .hint{position:absolute;right:3px;top:2px;font-size:11px;background:rgba(0,0,0,.7);color:#fff;padding:1px 5px;border-radius:999px;z-index:4}

    #summary{display:grid;gap:8px}

    .sum-title{font-weight:700;margin-bottom:4px}

    .sum-list{margin:0;padding-left:0;list-style:none;display:grid;gap:3px}

    .sum-item{display:flex;align-items:center;gap:6px;font-variant-numeric:tabular-nums}

    .sum-ico{width:16px;height:16px;display:inline-block;vertical-align:middle}

    /\* Overlay polys \*/

    #overlay polygon{

      stroke-linejoin:round; vector-effect:non-scaling-stroke; shape-rendering:geometricPrecision;

    }

    /\* Start modal \*/

    .modal{position:fixed;inset:0;background:rgba(0,0,0,.6);display:none;align-items:center;justify-content:center;z-index:1000}

    .modal.show{display:flex}

    .card{background:#fff;border-radius:12px;border:1px solid #ddd;padding:16px;max-width:620px;width:92%;box-shadow:0 12px 30px rgba(0,0,0,.25)}

    .sizes{display:grid;grid-template-columns:repeat(4,1fr);gap:8px;margin-top:8px}

    .swatches{display:flex;gap:8px;flex-wrap:wrap;margin-top:10px}

    .swatch{width:28px;height:28px;border-radius:6px;border:2px solid #ddd;cursor:pointer;box-shadow:inset 0 0 0 2px rgba(255,255,255,.35)}

    .swatch[data-sel="1"]{outline:3px solid #0003}

  </style>

</head>

<body>

  <header>

    <div class="paper-chip">Tour: <b id="turnLabel">Joueur</b></div>

    <div class="paper-chip">Or Joueur: <b id="gold1">0</b></div>

    <div class="paper-chip">Or Adversaires: <b id="gold2">0</b></div>

    <div class="controls">

      <button id="endTurn">Finir le tour</button>

      <button id="restart">Recommencer</button>

      <button id="menu">Menu</button>

      <span class="paper-chip">Seed: <b id="seedLabel">—</b></span>

    </div>

  </header>

  <div class="layout">

    <div id="sidebar">

      <div class="paper">

        <h3>Actions</h3>

        <div id="actions">

          <!-- Ville / construction -->

          <button id="buildCity" disabled>Construire Ville</button>

          <button id="buildTower" disabled>Bâtir Tour (3 or)</button>

          <!-- Recrutement dans la ville sélectionnée -->

          <button id="recFarmer"  disabled>Recruter Fermier</button>

          <button id="recMilitia" disabled>Recruter Milicien</button>

          <button id="recSoldier" disabled>Recruter Soldat</button>

          <!-- Améliorations dans la ville sélectionnée -->

          <button id="upFarmMil"  disabled>Améliorer Fermier → Milice</button>

          <button id="upMilSold"  disabled>Améliorer Milice → Soldat</button>

          <!-- Bateaux -->

          <button id="makeSloop" disabled>Sloop 1 mât (5 or)</button>

          <button id="makeBrig" disabled>Brigantin 2 mâts (10 or)</button>

          <button id="makeGalleon" disabled>Galion 3 mâts (20 or)</button>

          <button id="loadShip" disabled>Embarquer</button>

          <button id="disembark" disabled>Débarquer</button>

        </div>

      </div>

      <div class="paper">

        <h3>Résumé</h3>

        <div id="summary" style="font-size:13px;line-height:1.2"></div>

      </div>

      <div class="paper">

        <h3>Journal</h3>

        <div id="log"></div>

      </div>

    </div>

    <div id="board">

      <div id="stage">

        <svg id="terrain"></svg>

        <svg id="overlay"></svg>

        <div id="grid"></div>

      </div>

    </div>

  </div>

  <footer class="paper" style="margin:12px;border-radius:10px">

    Déplacements: Fermier 3 · Milicien 2 · Soldat 1 — Sloop 3 · Brigantin 2 · Galion 1.

  </footer>

  <!-- Start modal -->

  <div id="start" class="modal show">

    <div class="card">

      <h2>War-ixel — Nouvelle partie</h2>

      <p>Choisis <b>ta couleur</b>, la <b>taille</b>, le <b>nombre d’IA</b>, leurs <b>alliances</b> et la <b>difficulté</b> (1 novice → 4 chef de guerre), puis (optionnel) une <b>seed</b> :</p>

      <div class="swatches" id="swatches"></div>

      <div style="margin-top:10px; display:grid; gap:8px; grid-template-columns:repeat(2,1fr);">

        <label>IA (0–4)<br><input id="aiCount" type="number" min="0" max="4" value="1" style="width:100%"></label>

        <label>IA alliées entre elles ?<br><select id="aiAllied" style="width:100%"><option value="1" selected>Oui</option><option value="0">Non</option></select></label>

        <label>Difficulté<br>

          <select id="difficulty" style="width:100%">

            <option value="1">1 — Novice</option>

            <option value="2" selected>2 — Normal</option>

            <option value="3">3 — Stratège</option>

            <option value="4">4 — Chef de guerre</option>

          </select>

        </label>

        <div></div>

      </div>

      <div style="margin-top:10px; display:flex; gap:8px; align-items:center;">

        <label style="min-width:42px;">Seed</label>

        <input id="seedInput" type="text" placeholder="aléatoire"

               style="flex:1; padding:6px 8px; border:1px solid #ccc; border-radius:8px;">

        <button id="diceSeed" title="Seed aléatoire">🎲</button>

      </div>

      <div class="sizes" style="margin-top:8px">

        <button data-size="small">Petit</button>

        <button data-size="medium">Moyen</button>

        <button data-size="large">Grand</button>

        <button data-size="huge">Immense</button>

      </div>

    </div>

  </div>

  <script>

  (() => {

  /\* ==================== HEX GEO ==================== \*/

  const R = 34, HEX\_W = 2\*R, HEX\_H = Math.sqrt(3)\*R, STEP\_X = 1.5\*R, STEP\_Y = HEX\_H;

  function hexCenter(x,y){ return { cx: x\*STEP\_X+R, cy: y\*STEP\_Y + ((x&1)?HEX\_H/2:0) + HEX\_H/2 }; }

  function svgHexPoints(cx,cy,shrink=1){ const r=R\*shrink, h=Math.sqrt(3)\*r;

    const pts=[[cx-r/2,cy-h/2],[cx+r/2,cy-h/2],[cx+r,cy],[cx+r/2,cy+h/2],[cx-r/2,cy+h/2],[cx-r,cy]];

    return pts.map(p=>p.join(',')).join(' '); }

  /\* ==================== STATE / CONST ==================== \*/

  const UNITS={1:{key:1,name:'Fermier',cost:1,upkeep:0.5,move:3},2:{key:2,name:'Milicien',cost:2,upkeep:1,move:2},3:{key:3,name:'Soldat',cost:3,upkeep:1.5,move:1}};

  const SHIPS={sloop:{key:'sloop',name:'Sloop',cost:5,upkeep:2.5,move:3,water:'shallow',cap:1,hp:2,atk:1,masts:1},

               brig:{key:'brig',name:'Brigantin',cost:10,upkeep:5,move:2,water:'both',cap:2,hp:3,atk:1.5,masts:2},

               galleon:{key:'galleon',name:'Galion',cost:20,upkeep:10,move:1,water:'both',cap:3,hp:4,atk:2,masts:3}};

  const CAMO\_RANGE = 5;

  const CITY\_BASE\_COST=10, CITY\_SCALE=3;

  const SIZE\_PRESETS={small:{W:13,H:11},medium:{W:19,H:15},large:{W:25,H:19},huge:{W:33,H:25}};

  let W=13,H=11,S=[],selected=null,reach=[],reachTheo=[],turn=1,TOTAL\_PLAYERS=2,AI\_ALLIED=true,DIFFICULTY=2;

  const gold={};

  const PLAYER\_COLORS=[{n:"Émeraude",v:"#1f7a3e"},{n:"Bleu roi",v:"#1e3a8a"},{n:"Violet",v:"#6a1b9a"},{n:"Ambre",v:"#e39a1d"},{n:"Carmin",v:"#c62828"},{n:"Teal",v:"#00897b"}];

  let playerColor=PLAYER\_COLORS[1].v;

  const AI\_PALETTE=["#b71c1c","#6a1b9a","#e39a1d","#00897b"];

  function ownerColor(o){ if(o===1) return playerColor; return AI\_PALETTE[(o-2)%AI\_PALETTE.length]; }

  const boardEl=document.getElementById('board'), stageEl=document.getElementById('stage'), terrain=document.getElementById('terrain'),

        overlay=document.getElementById('overlay'), gridEl=document.getElementById('grid'), logEl=document.getElementById('log');

  /\* ==================== RNG / SEED ==================== \*/

  let CURRENT\_SIZE='small', CURRENT\_SEED=0;

  function mulberry32(a){return function(){let t=a+=0x6D2B79F5;t=Math.imul(t^t>>>15,t|1);t^=t+Math.imul(t^t>>>7,t|61);return((t^t>>>14)>>>0)/4294967296;};}

  let rand=Math.random;

  function hashStringToInt(s){ if(s==null) return 0; if(/^0x/i.test(s)) return parseInt(s,16)>>>0; if(/^\d+$/.test(s)) return (parseInt(s,10)>>>0);

    return Array.from(String(s)).reduce((h,ch)=>(Math.imul(31,h)+ch.charCodeAt(0))>>>0,0);}

  function setSeed(s){ const n=(typeof s==='string')?hashStringToInt(s):(s>>>0); CURRENT\_SEED=(n||1)>>>0; rand=mulberry32(CURRENT\_SEED); document.getElementById('seedLabel').textContent=CURRENT\_SEED; }

  /\* ==================== Colors ==================== \*/

  function hexToRgb(h){h=h.replace('#','');if(h.length===3)h=[...h].map(x=>x+x).join('');const num=parseInt(h,16);return{r:(num>>16)&255,g:(num>>8)&255,b:num&255};}

  function rgbToHex(r,g,b){const t=n=>n.toString(16).padStart(2,'0');return '#'+t(r)+t(g)+t(b);}

  function mix(c1,c2,p){const a=hexToRgb(c1),b=hexToRgb(c2);return rgbToHex(Math.round(a.r+(b.r-a.r)\*p),Math.round(a.g+(b.g-a.g)\*p),Math.round(a.b+(b.b-a.b)\*p));}

  function regionColor(owner){ if(owner===0) return '#dbdbdb'; const base=ownerColor(owner); return mix(base,'#ffffff',0.60); }

  /\* ==================== Noise / map gen ==================== \*/

  function hash(i,j){const s=Math.sin(i\*127.1+j\*311.7+(CURRENT\_SEED||0))\*43758.5453;return s-Math.floor(s);}

  function lerp(a,b,t){return a+(b-a)\*t;} function fade(t){return t\*t\*(3-2\*t);}

  function noise2(x,y){const i=Math.floor(x),j=Math.floor(y);const u=fade(x-i),v=fade(y-j);const a=hash(i,j),b=hash(i+1,j),c=hash(i,j+1),d=hash(i+1,j+1);return lerp(lerp(a,b,u),lerp(c,d,u),v);}

  function fbm(x,y,oct=4){let v=0,amp=0.5,freq=1;for(let o=0;o<oct;o++){v+=noise2(x\*freq,y\*freq)\*amp;freq\*=2;amp\*=0.5;}return v;}

  function inb(x,y){return x>=0&&y>=0&&x<W&&y<H;}

  function idx(x,y){return y\*W+x;}

  function cell(x,y){ if(!inb(x,y)) return undefined; return S[idx(x,y)]; }

  function neighbors(x,y){

    const out=[]; const odd=x&1;

    const E=[[-1,0],[+1,0],[0,-1],[0,+1],[-1,-1],[-1,+1]];

    const O=[[-1,0],[+1,0],[0,-1],[0,+1],[+1,-1],[+1,+1]];

    const dirs=odd?O:E;

    for(const[dx,dy] of dirs){ const nx=x+dx, ny=y+dy; if(inb(nx,ny)) out.push([nx,ny]); }

    return out;

  }

  function toCube(q,r){const x=q;const z=r-((q&1)?(q+1)/2:q/2);const y=-x-z;return{x,y,z};}

  function hexDist(a,b){const ac=toCube(a.x,a.y),bc=toCube(b.x,b.y);return Math.max(Math.abs(ac.x-bc.x),Math.abs(ac.y-bc.y),Math.abs(ac.z-bc.z));}

  function genMap(sizeKey, aiCount=1, allied=true, difficulty=2){

    AI\_ALLIED=!!+allied; DIFFICULTY=+difficulty||2; TOTAL\_PLAYERS=1+(+aiCount||0);

    const p=SIZE\_PRESETS[sizeKey]||SIZE\_PRESETS.small; W=p.W; H=p.H; S.length=0;

    for(let y=0;y<H;y++)for(let x=0;x<W;x++)S.push({x,y,terrain:'land',owner:0,structure:'none',unit:null,ship:null,e:0,shore:false});

    for(const c of S){

      const sx=c.x/W\*6, sy=c.y/H\*6; const elev=fbm(sx,sy,5), humid=fbm(sx+20,sy-13,4); c.e=elev;

      const edge=Math.min(c.x,W-1-c.x,c.y,H-1-c.y)/Math.min(W,H);

      const coastBias=(edge<0.08)?-0.2:0; const seaLevel=0.50+coastBias;

      if(elev<seaLevel-0.12) c.terrain='deep';

      else if(elev<seaLevel) c.terrain='shallow';

      else c.terrain=(humid>0.53?'forest':'land');

    }

    for(const c of S) if(c.terrain==='deep'&&neighbors(c.x,c.y).some(([nx,ny])=>['land','forest'].includes(cell(nx,ny)?.terrain))) c.terrain='shallow';

    for(const c of S) c.shore=((c.terrain==='land'||c.terrain==='forest') && neighbors(c.x,c.y).some(([nx,ny])=> ['shallow','deep'].includes(cell(nx,ny)?.terrain)));

    const starts=[]; starts.push(findNearestLandOnly(1,1));

    for(let k=0;k<aiCount;k++){const p2=randomFarLandOnly(starts,Math.floor(Math.min(W,H)/2.2))||randomFarLandOnly(starts,6)||findNearestLandOnly(W-2,H-2); starts.push(p2);}

    for(let i=0;i<starts.length;i++){ const o=i+1, p3=starts[i]; cell(p3.x,p3.y).structure='city'; cell(p3.x,p3.y).owner=o; spawnUnit(p3.x,p3.y,{owner:o,type:1,level:1},true); gold[o]=2; }

    turn=1; recalcMoved(false);

  }

  function findNearestLandOnly(sx,sy){let best={x:sx,y:sy},bestD=1e9;for(const c of S){if(c.terrain==='land'){const d=Math.abs(c.x-sx)+Math.abs(c.y-sy);if(d<bestD){bestD=d;best={x:c.x,y:c.y};}}}return best;}

  function randomFarLandOnly(points,minD){const candidates=S.filter(c=>(c.terrain==='land')&&c.structure==='none');for(let t=0;t<800;t++){const c=candidates[Math.floor(rand()\*candidates.length)];if(!c)break;if(points.every(p=>hexDist(c,p)>=minD))return{x:c.x,y:c.y};}return null;}

  function spawnUnit(x,y,u,moved=true){const c=cell(x,y);if(!c||c.unit||c.ship)return false;c.unit={owner:u.owner,type:u.type,level:u.level||1,moved};c.owner=u.owner;return true;}

  function recalcMoved(flag){for(const c of S){if(c.unit)c.unit.moved=flag;if(c.ship)c.ship.moved=flag;}}

  /\* ==================== Economie / résumé ==================== \*/

  function income(o){let inc=0;for(const c of S){if(c.owner===o&&(c.terrain==='land'||c.terrain==='forest'))inc++; if(c.owner===o&&c.structure==='city')inc+=2;} return inc;}

  function upkeep(o){let u=0;for(const c of S){if(c.unit&&c.unit.owner===o)u+=UNITS[c.unit.type].upkeep\*(c.unit.level||1); if(c.ship&&c.ship.owner===o)u+=SHIPS[c.ship.kind].upkeep;}return u;}

  function cityCost(o){const owned=S.filter(c=>c.structure==='city'&&c.owner===o).length-1;return CITY\_BASE\_COST + CITY\_SCALE\*Math.max(0,owned);}

  function countSummary(o){const units={1:0,2:0,3:0}, ships={sloop:0,brig:0,galleon:0}; for(const c of S){if(c.unit&&c.unit.owner===o)units[c.unit.type]+=(c.unit.level||1); if(c.ship&&c.ship.owner===o)ships[c.ship.kind]++;} return{units,ships};}

  function iconUnitSmall(t){const s='<svg class="sum-ico" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2.6" stroke-linecap="round" stroke-linejoin="round">'; if(t===1)return s+'<path d="M6 3 L6 15"/><path d="M3 7 L9 7"/><path d="M9 4 L9 8"/><path d="M6 15 L6 21"/><path d="M12 14 C12 12, 15 12, 15 14 C15 16, 12 16,12 14" />'+'</svg>'; if(t===2)return s+'<circle cx="12" cy="12" r="5.5"/><path d="M4 20 L20 4" />'+'</svg>'; return s+'<path d="M6 10 C6 6, 18 6, 18 10"/><path d="M6 10 L18 10"/><path d="M12 11 L12 20"/><path d="M9 20 L15 20"/>'+'</svg>'; }

  function iconShipSmall(){return `<svg class="sum-ico" viewBox="0 0 24 24"><path d="M3 16 H21 L18 20 H6 Z" fill="currentColor"/></svg>`;}

  function renderSummary(){const box=document.getElementById('summary'); if(!box)return; const me=countSummary(1); const upk=upkeep(1).toFixed(1);

    const army=[{k:1,label:`Fermier: ${me.units[1]} (loyer ${UNITS[1].upkeep} po/u)`},{k:2,label:`Milicien: ${me.units[2]} (loyer ${UNITS[2].upkeep} po/u)`},{k:3,label:`Soldat: ${me.units[3]} (loyer ${UNITS[3].upkeep} po/u)`}].map(u=>`<li class="sum-item">${iconUnitSmall(u.k)} ${u.label}</li>`).join('');

    const fleet=[{k:'sloop',label:`Sloop: ${me.ships.sloop} (loyer ${SHIPS.sloop.upkeep})`},{k:'brig',label:`Brigantin: ${me.ships.brig} (loyer ${SHIPS.brig.upkeep})`},{k:'galleon',label:`Galion: ${me.ships.galleon} (loyer ${SHIPS.galleon.upkeep})`}].map(s=>`<li class="sum-item">${iconShipSmall()} ${s.label}</li>`).join('');

    box.innerHTML=`<div class="sum-title">Armée</div><ul class="sum-list">${army}</ul><div class="sum-title" style="margin-top:6px">Flotte</div><ul class="sum-list">${fleet}</ul><div><b>Loyer total par tour : ${upk} or</b></div>`;}

  /\* ==================== Règles ==================== \*/

  function isEnemy(a,b){ if(a===b) return false; if(AI\_ALLIED&&a>1&&b>1) return false; return true; }

  function beats(att,def){ if(att===def) return 0; return (att>def)?1:-1; }

  /\* Forêt : fermier/milice coût x2 ; soldat interdit \*/

  function moveCostUnit(type, from, to, owner){

    if (!to) return Infinity;

    if (to.terrain === 'shallow' || to.terrain === 'deep') return Infinity;

    if (to.structure === 'tower' && isEnemy(owner, to.owner)) return Infinity;

    if (to.terrain === 'forest'){

      if (type === 1 || type === 2) return 2;

      return Infinity;

    }

    return 1;

  }

  function passableForUnit(from, to, owner, type){

    if (!to) return false;

    if (to.terrain === 'shallow' || to.terrain === 'deep') return false;

    if (to.terrain === 'forest' && !(type === 1 || type === 2)) return false;

    if (to.structure === 'tower' && isEnemy(owner, to.owner)) return false;

    if (to.unit && !isEnemy(owner, to.unit.owner) && to.unit.type !== type) return false;

    return true;

  }

  function passableForShip(from,to,owner,kind){

    if(!to) return false;

    if(to.ship) return false;

    if(to.structure==='tower'&&isEnemy(owner,to.owner)) return false;

    const rule=SHIPS[kind].water;

    if(rule==='shallow'&&to.terrain!=='shallow') return false;

    if(rule==='both'&&!(to.terrain==='shallow'||to.terrain==='deep')) return false;

    return true;

  }

  /\* Portées avec coût (Dijkstra) + annexion virtuelle \*/

  function reachableFrom(x, y, kind, param){

    const start = cell(x, y);

    const out = [];

    // --- Navires : inchangé (1 pas = 1 point)

    if (kind === 'ship'){

      const maxStep = SHIPS[start.ship.kind].move;

      const seen = new Map();

      const q = [[x, y, 0]];

      seen.set(idx(x,y), 0);

      while (q.length){

        const [cx, cy, st] = q.shift();

        const here = cell(cx, cy);

        for (const [nx, ny] of neighbors(cx, cy)){

          const there = cell(nx, ny);

          if (st + 1 > maxStep) continue;

          if (!passableForShip(here, there, start.ship.owner, start.ship.kind)) continue;

          const k = idx(nx, ny);

          if (seen.has(k) && seen.get(k) <= st + 1) continue;

          seen.set(k, st + 1);

          out.push([nx, ny]);

          q.push([nx, ny, st + 1]);

        }

      }

      return out;

    }

    // --- Unités terrestres : Dijkstra + annexion simulée

    const type   = start.unit.type;

    const owner  = start.unit.owner;

    const budget = UNITS[type].move;

    const virtOwned = new Set();

    for (const c of S) if (c.owner === owner) virtOwned.add(idx(c.x, c.y));

    const canAnnexVirt = (qx, qy) =>

      neighbors(qx, qy).some(([ax, ay]) => virtOwned.has(idx(ax, ay)));

    const dist   = new Map();

    const open   = [];

    const popMin = (arr) => { let k=0; for(let i=1;i<arr.length;i++) if(arr[i][2]<arr[k][2]) k=i; return arr.splice(k,1)[0]; };

    dist.set(idx(x,y), 0);

    open.push([x, y, 0]);

    while (open.length){

      const [cx, cy, costHere] = popMin(open);

      const here = cell(cx, cy);

      for (const [nx, ny] of neighbors(cx, cy)){

        const there = cell(nx, ny);

        // 1) Règles de terrain / tours / forêts / unité alliée d'un autre type

        let allowed = passableForUnit(here, there, owner, type);

        // Autoriser un combat (si terrain OK) même si passableForUnit dit non

        if (!allowed && there?.unit && isEnemy(owner, there.unit.owner)){

          allowed = (moveCostUnit(type, here, there, owner) < Infinity);

        }

        if (!allowed) continue;

        // 2) Règle d'annexion : neutre/ennemi ok seulement si adjacent au territoire virtuel

        if (there.owner !== owner){

          if (there.owner === 0){

            if (!canAnnexVirt(nx, ny)) continue;

          } else if (isEnemy(owner, there.owner)){

            if (!canAnnexVirt(nx, ny)) continue;

          }

        }

        // 3) Coût de pas

        const step = moveCostUnit(type, here, there, owner);

        if (step === Infinity) continue;

        const newCost = costHere + step;

        if (newCost > budget) continue;

        const k = idx(nx, ny);

        if (dist.has(k) && dist.get(k) <= newCost) continue;

        dist.set(k, newCost);

        out.push([nx, ny]);

        // annexion virtuelle d'une case neutre atteinte

        if (there.owner !== owner && there.owner === 0){

          virtOwned.add(k);

        }

        // 4) Arrêts naturels

        const stopHere =

          (there.unit && isEnemy(owner, there.unit.owner)) ||

          (there.owner > 0 && isEnemy(owner, there.owner)) ||

          (there.unit && !isEnemy(owner, there.unit.owner) && there.unit.type === type);

        if (!stopHere){

          open.push([nx, ny, newCost]);

        }

      }

    }

    return out;

  }

  function canAnnex(owner,x,y){ return neighbors(x,y).some(([nx,ny])=>cell(nx,ny)?.owner===owner); }

  /\* Camouflage (pour le joueur) \*/

  function enemyHiddenForPlayer(t){

    if(!t) return false;

    const hasEnemy=(t.unit&&isEnemy(1,t.unit.owner))||(t.structure!=='none'&&isEnemy(1,t.owner));

    if(!hasEnemy) return false;

    if(t.terrain!=='forest') return false;

    for(const c of S){ if(c.unit&&c.unit.owner===1){ if(hexDist(c,t)<=CAMO\_RANGE) return false; } }

    return true;

  }

  /\* Recrutement progressif + upgrades \*/

  function ownedUnitsOfType(owner,type){let n=0; for(const c of S) if(c.unit&&c.unit.owner===owner&&c.unit.type===type) n+=(c.unit.level||1); return n;}

  function recruitCost(type,owner){ return UNITS[type].cost + Math.floor(ownedUnitsOfType(owner,type)/5); }

  function upgradeCost(fromType,toType){ if(fromType===1&&toType===2) return 2; if(fromType===2&&toType===3) return 3; return 99; }

  /\* ==================== Dessin ==================== \*/

  function drawTerrain(){

    const Wpx=STEP\_X\*(W-1)+HEX\_W, Hpx=STEP\_Y\*(H-1)+HEX\_H+HEX\_H/2;

    stageEl.style.width=Wpx+'px'; stageEl.style.height=Hpx+'px';

    terrain.setAttribute('width',Wpx); terrain.setAttribute('height',Hpx); terrain.setAttribute('viewBox',`0 0 ${Wpx} ${Hpx}`); terrain.innerHTML='';

    const defs=document.createElementNS(terrain.namespaceURI,'defs');

    const waves=(id,stroke,opacity)=>{const p=document.createElementNS(terrain.namespaceURI,'pattern');p.id=id;p.setAttribute('patternUnits','userSpaceOnUse');p.setAttribute('width','12');p.setAttribute('height','10');const path=document.createElementNS(terrain.namespaceURI,'path');path.setAttribute('d','M0 6 Q3 8 6 6 T12 6');path.setAttribute('fill','none');path.setAttribute('stroke',stroke);path.setAttribute('stroke-width','1.2');path.setAttribute('opacity',opacity);p.appendChild(path);defs.appendChild(p);};

    waves('wavesShallow','#2b6da3',0.40); waves('wavesDeep','#9ecbff',0.22);

    const tree=document.createElementNS(terrain.namespaceURI,'pattern'); tree.id='treePattern'; tree.setAttribute('patternUnits','userSpaceOnUse'); tree.setAttribute('width','14'); tree.setAttribute('height','14');

    const g=document.createElementNS(terrain.namespaceURI,'g'); const tri=document.createElementNS(terrain.namespaceURI,'path'); tri.setAttribute('d','M7 2 L2 9 L12 9 Z'); tri.setAttribute('fill','rgba(34,85,34,.55)');

    const trunk=document.createElementNS(terrain.namespaceURI,'path'); trunk.setAttribute('d','M7 9 L7 12'); trunk.setAttribute('stroke','rgba(60,40,20,.7)'); trunk.setAttribute('stroke-width','2'); g.appendChild(tri); g.appendChild(trunk); tree.appendChild(g); defs.appendChild(tree);

    terrain.appendChild(defs);

    const gT=document.createElementNS(terrain.namespaceURI,'g'); terrain.appendChild(gT);

    for (let y=0; y<H; y++) for (let x=0; x<W; x++){

      const c = cell(x,y);

      if (!c) continue; // sûreté lorsque S n'est pas prêt

      const {cx,cy} = hexCenter(x,y);

      const poly=document.createElementNS(terrain.namespaceURI,'polygon');

      poly.setAttribute('points', svgHexPoints(cx,cy));

      let fill;

      if (c.terrain==='deep')    fill=getComputedStyle(document.documentElement).getPropertyValue('--deep').trim();

      else if (c.terrain==='shallow') fill=getComputedStyle(document.documentElement).getPropertyValue('--shallow').trim();

      else {

        const base=regionColor(c.owner||0);

        fill=(c.terrain==='forest') ? mix(base,'#6cab6c',0.22) : base;

      }

      poly.setAttribute('fill',fill);

      poly.setAttribute('stroke',fill);

      poly.setAttribute('stroke-width','1');

      poly.setAttribute('stroke-linejoin','round');

      poly.setAttribute('vector-effect','non-scaling-stroke');

      poly.setAttribute('shape-rendering','geometricPrecision');

      gT.appendChild(poly);

      if (c.terrain==='shallow' || c.terrain==='deep'){

        const top=document.createElementNS(terrain.namespaceURI,'polygon');

        top.setAttribute('points', svgHexPoints(cx,cy,0.992));

        top.setAttribute('fill', `url(#${c.terrain==='shallow'?'wavesShallow':'wavesDeep'})`);

        top.setAttribute('opacity','1'); gT.appendChild(top);

      } else if (c.terrain==='forest'){

        const ov=document.createElementNS(terrain.namespaceURI,'polygon');

        ov.setAttribute('points', svgHexPoints(cx,cy,0.992));

        ov.setAttribute('fill','url(#treePattern)');

        ov.setAttribute('opacity','1'); gT.appendChild(ov);

      }

    }

    overlay.setAttribute('width',Wpx); overlay.setAttribute('height',Hpx); overlay.setAttribute('viewBox',`0 0 ${Wpx} ${Hpx}`);

    gridEl.style.width=Wpx+'px'; gridEl.style.height=Hpx+'px';

  }

  function drawOverlay(){

    overlay.innerHTML=''; const gTheo=document.createElementNS(overlay.namespaceURI,'g'), gReach=document.createElementNS(overlay.namespaceURI,'g'); overlay.appendChild(gTheo); overlay.appendChild(gReach);

    const drawList=(arr,group,fill)=>{ for(const [x,y] of arr){ const {cx,cy}=hexCenter(x,y); const p=document.createElementNS(overlay.namespaceURI,'polygon'); p.setAttribute('points',svgHexPoints(cx,cy,0.985)); p.setAttribute('fill',fill); p.setAttribute('stroke',fill); p.setAttribute('stroke-width','1'); p.setAttribute('stroke-linejoin','round'); group.appendChild(p);} };

    drawList(reachTheo,gTheo,getComputedStyle(document.documentElement).getPropertyValue('--theo-fill').trim());

    drawList(reach,gReach,getComputedStyle(document.documentElement).getPropertyValue('--reach-fill').trim());

  }

  /\* ==================== Render ==================== \*/

  function render(){

    drawTerrain();

    // --- Pas de carte encore générée (modale ouverte) → évite toute lecture de S

    if (S.length !== W \* H){

      overlay.innerHTML = '';

      gridEl.innerHTML  = '';

      updateHUD();

      return;

    }

    // Portées

    reach = []; reachTheo = [];

    if (selected){

      const c = cell(selected.x, selected.y);

      if (selected.kind === 'unit' && c?.unit){

        reachTheo = reachableFrom(selected.x, selected.y, 'unit', c.unit.type);

        reach     = c.unit.moved ? [] : reachTheo.slice();

      }

      if (selected.kind === 'ship' && c?.ship){

        reachTheo = reachableFrom(selected.x, selected.y, 'ship', c.ship.kind);

        reach     = c.ship.moved ? [] : reachTheo.slice();

      }

    }

    drawOverlay();

    // Grille interactive

    gridEl.innerHTML = '';

    for (let y=0; y<H; y++) for (let x=0; x<W; x++){

      const c = cell(x,y);

      if (!c) continue; // sûreté

      const {cx,cy} = hexCenter(x,y);

      const d = document.createElement('div');

      d.className = 'hex';

      d.style.left = cx+'px'; d.style.top = cy+'px';

      d.style.width = HEX\_W+'px'; d.style.height = HEX\_H+'px';

      d.style.clipPath='polygon(50% 0%, 100% 25%, 100% 75%, 50% 100%, 0% 75%, 0% 25%)';

      d.dataset.x = x; d.dataset.y = y;

      d.addEventListener('click', onCellClick);

      // Frontière (avec garde stricte)

      const isLand = (c.terrain === 'land' || c.terrain === 'forest');

      const hasFrontier = isLand && neighbors(x,y).some(([nx,ny])=>{

        const n = cell(nx,ny);

        if (!n) return false;

        const nLand = (n.terrain === 'land' || n.terrain === 'forest');

        return nLand && n.owner !== c.owner;

      });

      if (hasFrontier){

        d.classList.add('frontier');

        if (c.owner === 1) d.classList.add('owned1');

        else if (c.owner === 2) d.classList.add('owned2');

      }

      if (selected && selected.x===x && selected.y===y) d.classList.add('selected');

      const hidden = enemyHiddenForPlayer(c);

      if (c.structure === 'city' && !hidden){

        const v=document.createElement('div'); v.className='city';

        if (c.owner===1 && c.terrain==='forest') v.classList.add('camo');

        v.appendChild(svgCastle()); d.appendChild(v);

        const h=document.createElement('div'); h.className='hint'; h.textContent='+2'; d.appendChild(h);

      }

      if (c.structure === 'tower' && !hidden){

        const t=document.createElement('div'); t.className='tower';

        if (c.owner===1 && c.terrain==='forest') t.classList.add('camo');

        t.appendChild(svgRook()); d.appendChild(t);

      }

      if (c.unit && !hidden){

        const u=document.createElement('div'); u.className='badge';

        u.style.background=ownerColor(c.unit.owner);

        if (c.unit.owner===1 && c.terrain==='forest') u.classList.add('camo');

        u.appendChild(svgIconUnit(c.unit.type)); d.appendChild(u);

      }

      if (c.ship){

        const s=document.createElement('div'); s.className='ship';

        s.style.background=ownerColor(c.ship.owner);

        s.appendChild(svgShip(c.ship.kind));

        const h=document.createElement('div'); h.className='hint';

        h.textContent=`${(c.ship.cargo?.reduce((a,u)=>a+(u.level||1),0)||0)}/${SHIPS[c.ship.kind].cap}`;

        d.appendChild(s); d.appendChild(h);

      }

      gridEl.appendChild(d);

    }

    updateHUD();

    updateActionsPanel();

    renderSummary();

    centerBoard();

  }

  function centerBoard(){ const extraX=Math.max(0,(stageEl.scrollWidth-boardEl.clientWidth)/2); const extraY=Math.max(0,(stageEl.scrollHeight-boardEl.clientHeight)/2); boardEl.scrollLeft=extraX; boardEl.scrollTop=extraY; }

  /\* ==================== SVG helpers ==================== \*/

  function svgIconUnit(type){const s=elSVG('0 0 24 24'); if(type===1){path(s,'M6 3 L6 15',2.6);path(s,'M3 7 L9 7',2.6);path(s,'M9 4 L9 8',2.6);path(s,'M6 15 L6 21',2.6);path(s,'M12 14 C12 12, 15 12, 15 14 C15 16, 12 16,12 14',2.2);return s;} if(type===2){circle(s,12,12,5.5,2.6);path(s,'M4 20 L20 4',2.6);return s;} path(s,'M6 10 C6 6, 18 6, 18 10',2.6);path(s,'M6 10 L18 10',2.6);path(s,'M12 11 L12 20',2.6);path(s,'M9 20 L15 20',2.6);return s;}

  function svgShip(kind){const s=elSVG('0 0 48 28'); pathC(s,'M4 18 L44 18 L38 24 L10 24 Z',1.5,'#444'); line(s,20,6,20,18,2); if(kind!=='sloop') line(s,28,6,28,18,2); return s;}

  function svgCastle(){const s=elSVG('0 0 24 16','castle'); pathW(s,'M2 6 L22 6 L22 14 L2 14 Z',1.6); pathW(s,'M2 6 L2 3 L5 3 L5 6 M8 6 L8 3 L11 3 L11 6 M14 6 L14 3 L17 3 L17 6 M20 6 L20 3 L22 3 L22 6',1.6); pathW(s,'M10 14 L10 10 L14 10 L14 14',1.6); return s;}

  function svgRook(){const s=elSVG('0 0 24 24'); pathFill(s,'M5 20 L19 20 L19 18 L17 16 L17 11 L18 10 L18 7 L15 7 L15 9 L12 9 L12 7 L9 7 L9 9 L6 9 L6 7 L5 7 L5 10 L6 11 L6 16 L5 18 Z','#777','#555',1.2); return s;}

  function elSVG(vb,cls){const s=document.createElementNS('http://www.w3.org/2000/svg','svg'); s.setAttribute('viewBox',vb); s.classList.add('icon'); if(cls) s.classList.add(cls); return s;}

  function path(s,d,sw=2){const p=document.createElementNS('http://www.w3.org/2000/svg','path'); p.setAttribute('d',d); p.setAttribute('fill','none'); p.setAttribute('stroke','#fff'); p.setAttribute('stroke-width',sw); p.setAttribute('stroke-linecap','round'); p.setAttribute('stroke-linejoin','round'); s.appendChild(p);}

  function pathW(s,d,sw=2){const p=document.createElementNS('http://www.w3.org/2000/svg','path'); p.setAttribute('d',d); p.setAttribute('fill','none'); p.setAttribute('stroke','#fff'); p.setAttribute('stroke-width',sw); p.setAttribute('stroke-linecap','round'); p.setAttribute('stroke-linejoin','round'); s.appendChild(p);}

  function pathC(s,d,sw=2,fill='#444'){const p=document.createElementNS('http://www.w3.org/2000/svg','path'); p.setAttribute('d',d); p.setAttribute('fill',fill); p.setAttribute('stroke','#eee'); p.setAttribute('stroke-width',sw); p.setAttribute('stroke-linejoin','round'); s.appendChild(p);}

  function pathFill(s,d,fill='#777',stroke='#555',sw=1){const p=document.createElementNS('http://www.w3.org/2000/svg','path'); p.setAttribute('d',d); p.setAttribute('fill',fill); p.setAttribute('stroke',stroke); p.setAttribute('stroke-width',sw); p.setAttribute('stroke-linejoin','round'); s.appendChild(p);}

  function circle(s,cx,cy,r,sw=2){const c=document.createElementNS('http://www.w3.org/2000/svg','circle'); c.setAttribute('cx',cx); c.setAttribute('cy',cy); c.setAttribute('r',r); c.setAttribute('fill','none'); c.setAttribute('stroke','#fff'); c.setAttribute('stroke-width',sw); s.appendChild(c);}

  function line(s,x1,y1,x2,y2,sw=2){const l=document.createElementNS('http://www.w3.org/2000/svg','line'); l.setAttribute('x1',x1); l.setAttribute('y1',y1); l.setAttribute('x2',x2); l.setAttribute('y2',y2); l.setAttribute('stroke','#eee'); l.setAttribute('stroke-width',sw); s.appendChild(l);}

  /\* ==================== HUD & LOG ==================== \*/

  function updateHUD(){ document.getElementById('gold1').textContent=Math.floor(gold[1]||0); let sum=0; for(let o=2;o<=TOTAL\_PLAYERS;o++) sum+=Math.floor(gold[o]||0); document.getElementById('gold2').textContent=sum; document.getElementById('turnLabel').textContent=(turn===1?'Joueur':`IA ${turn-1}`); }

  function addLog(s,d=null,owner=null){ const e=document.createElement('div'); e.className='entry'; if(owner) e.dataset.owner=owner; e.innerHTML=`<div class="summary">• ${s}</div><div class="detail">${d?d:s}</div>`; e.querySelector('.summary').onclick=()=>e.classList.toggle('open'); logEl.insertAdjacentElement('afterbegin',e); }

  function log(s,o=null){ addLog(s,null,o); }

  /\* ==================== Sélection / clics ==================== \*/

  function onCellClick(e){

    const x=+e.currentTarget.dataset.x, y=+e.currentTarget.dataset.y, c=cell(x,y);

    if (selected && reach.some(([mx,my]) => mx===x && my===y)) {

      if (selected.kind === 'unit')      moveUnitPlayer(selected.x, selected.y, x, y);

      else if (selected.kind === 'ship') moveShipPlayer(selected.x, selected.y, x, y);

      const here=cell(x,y);

      selected={x,y,kind:(here?.ship?'ship':(here?.unit?'unit':(here?.structure==='city'&&here?.owner===1?'city':'empty')))};

      render(); return;

    }

    if(turn!==1){ selected=null; render(); return; }

    // Toggle ville/unité quand une unité alliée est sur la ville

    if(c?.structure==='city' && c.owner===1 && c.unit && c.unit.owner===1){

      if(selected && selected.x===x && selected.y===y){

        selected = (selected.kind==='unit') ? {x,y,kind:'city'} : {x,y,kind:'unit'};

      }else{

        selected = {x,y,kind:'unit'};

      }

      render(); return;

    }

    if(c?.unit && c.unit.owner===1) selected={x,y,kind:'unit'};

    else if(c?.ship && c.ship.owner===1) selected={x,y,kind:'ship'};

    else if(c?.structure==='city' && c.owner===1) selected={x,y,kind:'city'};

    else if((c?.terrain==='land') && c.owner===1 && !c.unit && !c.ship) selected={x,y,kind:'empty'};

    else selected=null;

    render();

  }

  /\* ==================== Déplacements / actions ==================== \*/

  function unitName(t){ return (UNITS[t]?.name||'Unité').toLowerCase(); }

  // moveUnit générique (IA et joueur). Option forcedOwner pour verrouiller côté UI.

  function moveUnit(sx, sy, dx, dy, forcedOwner = null){

    const src = cell(sx, sy), dst = cell(dx, dy);

    if (!src?.unit || src.unit.moved) return;

    // Si forcedOwner est fourni (UI), s'assurer que c'est bien le bon proprio

    if (forcedOwner !== null && src.unit.owner !== forcedOwner) return;

    const me = src.unit.owner;

    // Interdit par le terrain ?

    if (moveCostUnit(src.unit.type, src, dst, me) === Infinity) return;

    // Fusion alliée (même type)

    if (dst?.unit && !isEnemy(me, dst.unit.owner) && dst.unit.type === src.unit.type){

      dst.unit.level = (dst.unit.level || 1) + (src.unit.level || 1);

      dst.unit.moved = true;

      src.unit = null;

      dst.owner = me;

      addLog(`Fusion en ${dx},${dy}`, `Pile de ${UNITS[dst.unit.type].name} ×${dst.unit.level}`, me);

      return;

    }

    // Combat (via beats)

    if (dst?.unit && isEnemy(me, dst.unit.owner)){

      const r = beats(src.unit.type, dst.unit.type);

      if (r === 1){

        if (!canAnnex(me, dx, dy)) { addLog("Impossible d'annexer","Il faut être adjacent à ton territoire.",me); return; }

        dst.unit = { ...src.unit, moved:true }; src.unit = null; dst.owner = me;

        addLog(`Victoire en ${dx},${dy}`, `Ton ${unitName(dst.unit.type)} l’emporte.`, me);

      } else if (r === -1){

        src.unit = null;

        addLog(`Défaite en ${dx},${dy}`, `Ton ${unitName(src.unit?.type)} perd le duel.`, me);

      } else {

        src.unit = null; dst.unit = null;

        addLog(`Égalité en ${dx},${dy}`, `Les deux unités périssent.`, me);

      }

      return;

    }

    // Mouvement / annexion

    if (!canAnnex(me, dx, dy)) { addLog("Impossible d'annexer","Il faut être adjacent à ton territoire.",me); return; }

    dst.unit = { ...src.unit, moved:true }; dst.owner = me; src.unit = null;

    addLog(`Prise de ${dx},${dy}`, 'La case passe sous ton contrôle.', me);

  }

  function moveShip(sx,sy,dx,dy){

    const src=cell(sx,sy), dst=cell(dx,dy);

    if(!src?.ship||src.ship.moved||dst?.ship) return;

    dst.ship={...src.ship,moved:true};

    src.ship=null;

  }

  // Wrappers "UI" (vérif tour + proprio) — utilisés par onCellClick

  function moveUnitPlayer(sx, sy, dx, dy){

    if (turn !== 1) return;

    const src = cell(sx, sy);

    if (!src?.unit || src.unit.owner !== 1) return;

    moveUnit(sx, sy, dx, dy, 1); // verrou joueur

  }

  function moveShipPlayer(sx, sy, dx, dy){

    if (turn !== 1) return;

    const src = cell(sx, sy);

    if (!src?.ship || src.ship.owner !== 1) return;

    moveShip(sx, sy, dx, dy);

  }

  const B=id=>document.getElementById(id);

  /\* Construction / bateaux \*/

  B('buildCity').onclick=()=>{ const c=cell(selected.x,selected.y); const cc=cityCost(1); if((gold[1]||0)<cc||c?.structure!=='none'||c?.terrain!=='land') return; c.structure='city'; c.owner=1; gold[1]-=cc; addLog('Ville fondée',`En ${selected.x},${selected.y} (coût ${cc} or)`,1); render(); };

  B('buildTower').onclick=()=>{ const c=cell(selected.x,selected.y); if((gold[1]||0)<3||c?.structure!=='none'||c?.terrain!=='land') return; c.structure='tower'; c.owner=1; gold[1]-=3; addLog('Tour bâtie',`En ${selected.x},${selected.y}`,1); render(); };

  B('makeSloop').onclick=()=>makeShip('sloop'); B('makeBrig').onclick=()=>makeShip('brig'); B('makeGalleon').onclick=()=>makeShip('galleon');

  B('loadShip').onclick=()=>loadShip(); B('disembark').onclick=()=>disembark();

  /\* Recrutement & upgrades (via panneau) \*/

  B('recFarmer').onclick = ()=>{ const s=selected; if(!s||s.kind!=='city') return; recruitAtCity(s.x,s.y,1); };

  B('recMilitia').onclick= ()=>{ const s=selected; if(!s||s.kind!=='city') return; recruitAtCity(s.x,s.y,2); };

  B('recSoldier').onclick= ()=>{ const s=selected; if(!s||s.kind!=='city') return; recruitAtCity(s.x,s.y,3); };

  B('upFarmMil').onclick = ()=>{ const s=selected; if(!s||s.kind!=='city') return; tryUpgradeAtCity(s.x,s.y,2); };

  B('upMilSold').onclick = ()=>{ const s=selected; if(!s||s.kind!=='city') return; tryUpgradeAtCity(s.x,s.y,3); };

  function updateActionsPanel(){

    const sel=selected?cell(selected.x,selected.y):null, myTurn=(turn===1);

    /\* Libellé coût ville dynamique \*/

    const cc=cityCost(1); const bc=document.getElementById('buildCity'); bc.textContent=`Construire Ville (${cc} or)`;

    const onUnit=myTurn&&sel&&selected.kind==='unit'&&sel.unit&&!sel.unit.moved;

    const coastal=onUnit&&neighbors(selected.x,selected.y).some(([nx,ny])=>{const t=cell(nx,ny);return !!t && (t.terrain==='shallow'||t.terrain==='deep') && !t.ship; });

    B('makeSloop').disabled=!(onUnit&&coastal&&(gold[1]||0)>=SHIPS.sloop.cost);

    B('makeBrig').disabled=!(onUnit&&coastal&&(gold[1]||0)>=SHIPS.brig.cost);

    B('makeGalleon').disabled=!(onUnit&&coastal&&(gold[1]||0)>=SHIPS.galleon.cost);

    const ownEmptyLand=myTurn&&sel&&selected.kind==='empty'&&sel.terrain==='land'&&sel.structure==='none';

    B('buildCity').disabled=!(ownEmptyLand&&(gold[1]||0)>=cc);

    B('buildTower').disabled=!(ownEmptyLand&&(gold[1]||0)>=3);

    const onShip=myTurn&&sel&&selected.kind==='ship'&&sel.ship&&!sel.ship.moved;

    B('loadShip').disabled=!(onShip && capacityLeft(sel)>0 && neighbors(selected.x,selected.y).some(([nx,ny])=>{const t=cell(nx,ny);return t?.unit && t.unit.owner===1 && (t.unit.level||1)<=capacityLeft(sel);} ) );

    B('disembark').disabled = !(onShip && (sel.ship.cargo?.length||0)>0 && neighbors(selected.x,selected.y).some(([nx,ny])=>{

      const t = cell(nx,ny);

      return !!t

        && (t.terrain==='land' || t.terrain==='forest')

        && !t.unit

        && !(t.structure==='tower' && isEnemy(1,t.owner))

        && (t.owner===0 || canAnnex(1,nx,ny)); // autoriser plage neutre

    }));

    /\* Recrutement/upgrade en ville (panneau) \*/

    const onCity=myTurn&&sel&&selected.kind==='city'&&sel.structure==='city'&&sel.owner===1;

    const occupied = !!(onCity && (sel.unit || sel.ship));

    const cF = recruitCost(1,1), cM = recruitCost(2,1), cS = recruitCost(3,1);

    document.getElementById('recFarmer').textContent  = `Recruter Fermier (${cF} or)`;

    document.getElementById('recMilitia').textContent = `Recruter Milicien (${cM} or)`;

    document.getElementById('recSoldier').textContent = `Recruter Soldat (${cS} or)`;

    B('recFarmer').disabled  = !(onCity && !occupied && (gold[1]||0)>=cF);

    B('recMilitia').disabled = !(onCity && !occupied && (gold[1]||0)>=cM);

    B('recSoldier').disabled = !(onCity && !occupied && (gold[1]||0)>=cS);

    // upgrades: uniquement si une unité du joueur est DANS la ville

    let canUpFM=false, canUpMS=false, labFM='Améliorer Fermier → Milice', labMS='Améliorer Milice → Soldat';

    if(onCity && sel.unit && sel.unit.owner===1){

      if(sel.unit.type===1){ const price=upgradeCost(1,2); labFM += ` (${price} or)`; canUpFM=(gold[1]||0)>=price; }

      if(sel.unit.type===2){ const price=upgradeCost(2,3); labMS += ` (${price} or)`; canUpMS=(gold[1]||0)>=price; }

    }

    document.getElementById('upFarmMil').textContent = labFM;

    document.getElementById('upMilSold').textContent = labMS;

    B('upFarmMil').disabled = !(onCity && sel?.unit?.owner===1 && sel.unit.type===1 && canUpFM);

    B('upMilSold').disabled = !(onCity && sel?.unit?.owner===1 && sel.unit.type===2 && canUpMS);

  }

  function capacityLeft(shipCellObj){ const ship=shipCellObj.ship||shipCellObj; return SHIPS[ship.kind].cap - (ship.cargo?.reduce((a,u)=>a+(u.level||1),0)||0); }

  function makeShip(kind){

    const u=cell(selected.x,selected.y);

    if(!u?.unit || u.unit.moved || u.unit.owner !== 1) return; // garde propriétaire

    const adj=neighbors(selected.x,selected.y).find(([nx,ny])=>{

      const t=cell(nx,ny);

      if(!t||t.ship) return false;

      const rule=SHIPS[kind].water;

      if(rule==='shallow'&&t.terrain!=='shallow') return false;

      if(rule==='both'&&!(t.terrain==='shallow'||t.terrain==='deep')) return false;

      return true;

    });

    if(!adj){addLog("Pas d'eau libre adjacente.",null,1);return;}

    if((gold[1]||0) < SHIPS[kind].cost){addLog("Pas assez d'or.",null,1);return;}

    gold[1]-=SHIPS[kind].cost; const [sx,sy]=adj; cell(sx,sy).ship={owner:1,kind,cargo:[],moved:true,hp:SHIPS[kind].hp}; addLog(`${SHIPS[kind].name} mis à l’eau`,`En ${sx},${sy} (capacité ${SHIPS[kind].cap})`,1); render();

  }

  function loadShip(){ const s=cell(selected.x,selected.y); if(!s?.ship||s.ship.moved) return; if(!s.ship.cargo) s.ship.cargo=[];

    const capLeft=capacityLeft(s);

    const pos=neighbors(selected.x,selected.y).find(([nx,ny])=>{ const t=cell(nx,ny); return t?.unit && t.unit.owner===1 && (t.unit.level||1)<=capLeft; });

    if(!pos){addLog("Aucune unité admissible à embarquer (capacité insuffisante).",null,1);return;}

    const [ux,uy]=pos; const u=cell(ux,uy); s.ship.cargo.push({...u.unit}); u.unit=null; addLog('Embarquement',`Depuis ${ux},${uy}`,1); render();

  }

  function disembark(){

    const s = cell(selected.x, selected.y);

    if (!s?.ship || s.ship.moved || !(s.ship.cargo && s.ship.cargo.length)) return;

    // autoriser le débarquement sur case neutre OU adjacente au territoire

    const spot = neighbors(selected.x, selected.y).find(([nx, ny]) => {

      const t = cell(nx, ny);

      return !!t

        && (t.terrain === 'land' || t.terrain === 'forest')

        && !t.unit

        && !(t.structure === 'tower' && isEnemy(1, t.owner))

        && (t.owner === 0 || canAnnex(1, nx, ny));

    });

    if (!spot) {

      addLog("Débarquement interdit : aucun emplacement valide (terre libre requise).", null, 1);

      return;

    }

    const [dx, dy] = spot;

    const t = cell(dx, dy);

    const u = s.ship.cargo.shift();

    t.unit = { ...u, moved: true };

    t.owner = 1;                // annexe la case en débarquant

    s.ship.moved = true;

    addLog('Débarquement', `En ${dx},${dy}`, 1);

    render();

  }

  /\* ====== Recrutement / Upgrades côté panneau ====== \*/

  function recruitAtCity(x,y,type){

    const c=cell(x,y);

    if(!c||c.structure!=='city'||c.owner!==1){ addLog("Sélectionne d'abord ta ville.",null,1); return; }

    if(c.unit || c.ship){ addLog("Recrutement impossible","La ville est occupée.",1); return; }

    const cost=recruitCost(type,1);

    if((gold[1]||0) < cost){ addLog("Pas assez d'or",`Coût ${cost} or`,1); return; }

    gold[1] -= cost;

    c.unit = { owner:1, type, level:1, moved:true };

    addLog(`Recrutement: ${UNITS[type].name}`, `En ${x},${y} (coût ${cost})`, 1);

    render();

  }

  function tryUpgradeAtCity(x,y,toTypeMaybe){

    const c=cell(x,y);

    if(!c||c.structure!=='city'||c.owner!==1||!c.unit||c.unit.owner!==1){ addLog("Aucune unité à améliorer dans la ville.",null,1); return; }

    let to = toTypeMaybe || (c.unit.type===1?2:(c.unit.type===2?3:3));

    if(c.unit.type===3){ addLog("Déjà au rang maximum.",null,1); return; }

    const cost=upgradeCost(c.unit.type,to);

    if((gold[1]||0)<cost){ addLog("Or insuffisant pour l'amélioration",`Coût ${cost} or`,1); return; }

    gold[1]-=cost; c.unit.type=to; c.unit.moved=true; addLog(`Amélioration: ${UNITS[to].name}`,`En ${x},${y} (coût ${cost})`,1); render();

  }

  /\* ==================== Fin de tour & IA ==================== \*/

  document.getElementById('endTurn').onclick=()=>endTurn();

  document.getElementById('restart').onclick=()=>{ setSeed(CURRENT\_SEED); const aiCount=+document.getElementById('aiCount').value||0; const allied=+document.getElementById('aiAllied').value; const diff=+document.getElementById('difficulty').value||2; genMap(CURRENT\_SIZE,aiCount,allied,diff); render(); addLog(`Relance de la carte (seed ${CURRENT\_SEED})`,null,1); };

  document.getElementById('menu').onclick=()=>showStart();

  function applyBankruptcy(owner){ const upk=upkeep(owner); if((gold[owner]||0)>=upk) return; for(const c of S){ if(c.unit&&c.unit.owner===owner)c.unit=null; if(c.ship&&c.ship.owner===owner)c.ship=null; } gold[owner]=Math.max(0,gold[owner]||0); addLog(`${owner===1?'Joueur':('IA '+(owner-1))} en banqueroute : armée dissoute`,`Loyer (${upk.toFixed(1)}) > or disponible.`,owner); }

  function endTurn(){

    const inc=income(turn); gold[turn]=(gold[turn]||0)+inc; const upk=upkeep(turn); gold[turn]-=upk;

    addLog(`Tour ${turn===1?'Joueur':'IA '+(turn-1)} : +${inc} or, -${upk.toFixed(1)} or de loyer`,null,turn); recalcMoved(false); selected=null;

    turn++; if(turn>TOTAL\_PLAYERS) turn=1; applyBankruptcy(turn);

    render(); checkVictory(); if(turn!==1) setTimeout(()=>aiTurn(turn),250);

  }

  function checkVictory(){ const aiAlive=S.some(c=>(c.unit&&c.unit.owner>1)||(c.structure==='city'&&c.owner>1)); const playerAlive=S.some(c=>(c.unit&&c.unit.owner===1)||(c.structure==='city'&&c.owner===1)); if(!aiAlive) setTimeout(()=>alert("Victoire !"),60); if(!playerAlive) setTimeout(()=>alert("Défaite…"),60); }

  function aiTurn(owner){

    const RESERVE=[2,4,6,10][DIFFICULTY-1]||4;

    // Recrutement en ville

    const vAI=S.filter(c=>c.structure==='city'&&c.owner===owner&&!c.unit&&!c.ship);

    for(const v of vAI){

      const tryRecruit=t=>{ const cost=recruitCost(t,owner), upk=UNITS[t].upkeep, inc=income(owner), nextGold=(gold[owner]||0)-cost+inc-(upkeep(owner)+upk);

        if((gold[owner]||0)-cost>=RESERVE && nextGold>=0){ v.unit={owner,type:t,level:1,moved:true}; gold[owner]-=cost; addLog(`IA ${owner-1} recrute un ${UNITS[t].name}`,null,owner); return true; } return false; };

      if(tryRecruit(3)) continue; if(tryRecruit(2)) continue; tryRecruit(1);

    }

    // Mise à l'eau (limite selon surface)

    const tilesAI=S.filter(c=>c.owner===owner&&(c.terrain==='land'||c.terrain==='forest')).length;

    const maxShips=Math.max(1,Math.floor(tilesAI/(DIFFICULTY>=3?6:8))); const curShips=S.filter(c=>c.ship&&c.ship.owner===owner).length;

    if(curShips<maxShips){

      const coastal=S.find(c=>c.unit&&c.unit.owner===owner&&neighbors(c.x,c.y).some(([nx,ny])=>{const t=cell(nx,ny);return !!t && t.terrain!=='land'&&!t.ship;}));

      if(coastal){

        const seaShallow=neighbors(coastal.x,coastal.y).find(([nx,ny])=>{const t=cell(nx,ny);return t?.terrain==='shallow'&&!t.ship;});

        const tryShip=seaShallow?'sloop':'brig', shipDef=SHIPS[tryShip], cost=shipDef.cost, upk=shipDef.upkeep, inc=income(owner);

        const nextGold=(gold[owner]||0)-cost+inc-(upkeep(owner)+upk);

        if((gold[owner]||0)-cost>=RESERVE && nextGold>=0){

          const pos=seaShallow?seaShallow:neighbors(coastal.x,coastal.y).find(([nx,ny])=>{const t=cell(nx,ny);return !!t && (t.terrain==='deep'||t.terrain==='shallow')&&!t.ship;});

          if(pos){ const [sx,sy]=pos; gold[owner]-=cost; cell(sx,sy).ship={owner,kind:tryShip,cargo:[{...coastal.unit}],moved:true,hp:shipDef.hp}; coastal.unit=null; addLog(`IA ${owner-1} met un ${shipDef.name.toLowerCase()} à l’eau`,null,owner); }

        }

      }

    }

    // Déplacements unités

    const units=S.filter(c=>c.unit&&c.unit.owner===owner);

    for(const u of units){

      if(u.unit.moved) continue;

      const tgt=nearestEnemyOrCity(u.x,u.y,owner);

      const opts=reachableFrom(u.x,u.y,'unit',u.unit.type);

      if(opts.length && tgt){

        opts.sort((a,b)=>hexDist({x:a[0],y:a[1]},tgt)-hexDist({x:b[0],y:b[1]},tgt));

        const [nx,ny]=opts[0];

        moveUnit(u.x,u.y,nx,ny); // IA appelle la version générique sans forcedOwner

      }

    }

    // Déplacements navires

    const boats=S.filter(c=>c.ship&&c.ship.owner===owner);

    for(const b of boats){

      if(b.ship.moved) continue;

      const shore=neighbors(b.x,b.y).find(([nx,ny])=>{

        const t=cell(nx,ny);

        return !!t && (t.terrain==='land'||t.terrain==='forest') && !t.unit &&

               !(t.structure==='tower'&&isEnemy(owner,t.owner)) &&

               (t.owner===0 || canAnnex(owner,nx,ny));

      }); // autoriser plage neutre

      if(shore && b.ship.cargo&&b.ship.cargo.length){ const [dx,dy]=shore; const t=cell(dx,dy); t.unit={...b.ship.cargo.shift(),moved:true}; t.owner=owner; b.ship.moved=true; addLog(`IA ${owner-1} débarque`,null,owner); continue; }

      const tgt=nearestPlayerCoast(owner); const opts=reachableFrom(b.x,b.y,'ship',b.ship.kind);

      if(tgt&&opts.length){ opts.sort((a,b2)=>hexDist({x:a[0],y:a[1]},tgt)-hexDist({x:b2[0],y:b2[1]},tgt)); const [nx,ny]=opts[0]; moveShip(b.x,b.y,nx,ny); }

    }

    render(); checkVictory(); endTurn();

  }

  function nearestEnemyOrCity(x,y,owner){ let best=null,bestD=1e9; for(const c of S){ if(((c.unit&&isEnemy(owner,c.unit.owner))||(c.structure==='city'&&isEnemy(owner,c.owner)))){ const d=hexDist({x,y},c); if(d<bestD){bestD=d;best=c;} } } return best; }

  function nearestPlayerCoast(owner){ let best=null,bestD=1e9; for(const c of S){ if((c.terrain==='shallow'||c.terrain==='deep')&&neighbors(c.x,c.y).some(([nx,ny])=>{const t=cell(nx,ny);return !!t && isEnemy(owner,t.owner) && (t.terrain==='land'||t.terrain==='forest');})){ const d=Math.abs(c.x-(W>>1))+Math.abs(c.y-(H>>1)); if(d<bestD){bestD=d;best=c;} } } return best; }

  /\* ==================== Start / init ==================== \*/

  function setPlayerColor(hex){ playerColor=hex; document.documentElement.style.setProperty('--my',playerColor); }

  function buildSwatches(){ const wrap=document.getElementById('swatches'); wrap.innerHTML=''; PLAYER\_COLORS.forEach((c,i)=>{ const b=document.createElement('div'); b.className='swatch'; b.style.background=c.v; b.dataset.sel=(i===1?'1':'0'); b.title=c.n; b.addEventListener('click',()=>{wrap.querySelectorAll('.swatch').forEach(s=>s.dataset.sel='0'); b.dataset.sel='1'; setPlayerColor(c.v); render();}); wrap.appendChild(b);}); setPlayerColor(PLAYER\_COLORS[1].v); }

  function showStart(){ document.getElementById('start').classList.add('show'); }

  function hideStart(){ document.getElementById('start').classList.remove('show'); }

  document.querySelectorAll('#start button[data-size]').forEach(b=>{

    b.addEventListener('click',()=>{

      const key=b.dataset.size, seedField=document.getElementById('seedInput'), seedText=(seedField?.value||'').trim();

      if(seedText==='') setSeed(Math.floor(Math.random()\*2\*\*31)>>>0); else setSeed(seedText);

      const aiCount=+document.getElementById('aiCount').value||0, allied=+document.getElementById('aiAllied').value, diff=+document.getElementById('difficulty').value||2;

      CURRENT\_SIZE=key; genMap(key,aiCount,allied,diff); hideStart(); render(); addLog(`Nouvelle partie (${key}, seed ${CURRENT\_SEED}, IA ${aiCount}, alliées=${!!+allied}, diff ${diff})`,null,1);

    });

  });

  document.getElementById('diceSeed').addEventListener('click',()=>{ const r=(Math.floor(Math.random()\*2\*\*31)>>>0); document.getElementById('seedInput').value=r; });

  window.addEventListener('DOMContentLoaded',()=>{

    buildSwatches(); setSeed(Math.floor(Math.random()\*2\*\*31)>>>0); const seedInput=document.getElementById('seedInput'); if(seedInput) seedInput.value=CURRENT\_SEED; showStart();

    setTimeout(()=>{ if(!document.getElementById('start').classList.contains('show')){ genMap('small',1,true,2); render(); } },120);

  });

  window.addEventListener('error',(e)=>{ const log=document.getElementById('log'); if(log) log.innerHTML=`<div style="color:#a00">Erreur JS: ${e.message}</div>`+log.innerHTML; });

  })(); /\* IIFE end \*/

  </script>

</body>

</html>