[HTML] Simulation RPG Combat Sample

- Battle Sample

Battle Sample

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<style>
*{box-sizing:border-box;margin:0;padding:0}
body{font-family:sans-serif;background:#1a1a2e;color:#eee;user-select:none;min-height:100vh}
#app{display:flex;flex-direction:column;align-items:center;padding:12px;gap:10px}
h1{font-size:18px;font-weight:500;color:#c9a96e;letter-spacing:2px}
#top{display:flex;gap:16px;align-items:flex-start;width:100%;max-width:720px}
#map-wrap{position:relative}
canvas{display:block;border:2px solid #444;border-radius:4px}
#sidebar{flex:1;min-width:180px;display:flex;flex-direction:column;gap:8px}
.panel{background:#16213e;border:1px solid #2a3a5e;border-radius:6px;padding:10px;font-
size:13px}
.panel h3{font-size:12px;color:#c9a96e;margin-bottom:6px;text-transform:uppercase;letter-
spacing:1px}
#unit-name{font-size:15px;font-weight:500;margin-bottom:4px}
.bar-row{display:flex;align-items:center;gap:6px;margin:3px 0}
.bar-label{width:28px;font-size:11px;color:#aaa}
.bar-bg{flex:1;height:8px;background:#2a2a3e;border-radius:4px;overflow:hidden}
.bar-fill{height:100%;border-radius:4px;transition:width .3s}
.hp-bar{background:#4caf50}
.mp-bar{background:#5599ff}
#log{height:120px;overflow-y:auto;font-size:12px;line-height:1.6;color:#bbb}
#log div{padding:1px 0;border-bottom:1px solid #1e2a40}
#bottom{display:flex;gap:8px;flex-wrap:wrap;justify-content:center}
button{padding:6px 14px;background:#1e3a5f;border:1px solid #3a5a8f;color:#c9d8f0;border-
radius:4px;cursor:pointer;font-size:13px;transition:background .15s}
button:hover{background:#2a4a7f}
button:disabled{opacity:.4;cursor:default}
button.danger{background:#5f1e1e;border-color:#8f3a3a;color:#f0c9c9}
button.danger:hover{background:#7f2a2a}
#phase{font-size:13px;color:#c9a96e;text-align:center}
#weapon-info{display:grid;grid-template-columns:1fr 1fr;gap:4px}
```

```
.wi{font-size:11px;padding:3px 5px;background:#0d1626;border-radius:3px;border-left:3px solid
#3a5a8f}
.wi b{color:#c9a96e}
</style>
</head>
<body>
<div id="app">
<h1>⚔ SRPG Simulator</h1>
<div id="top">
  <div id="map-wrap"><canvas id="c" width="400" height="400"></canvas></div>
  <div id="sidebar">
    <div class="panel" id="info-panel">
      <h3>Selected unit</h3>
      <div id="unit-name" style="color:#aaa">Select Unit</div>
      <div id="unit-stats"></div>
    </div>
    <div class="panel">
      <h3>Weapon Types</h3>
      <div id="weapon-info">
        <div class="wi"><b>□□Sword</b><br>Atk 15 Acc 90%<br>Range 1 Balanced</div>
        <div class="wi"><b>□□Bow</b><br>Atk 10 Acc 80%<br>Range 2 Distance</div>
        <div class="wi"><b>□□Spear</b><br>Atk 13 Acc 85%<br>Range 1 Counter+</div>
        <div class="wi"><b>□□Axe</b><br>Atk 20 Acc 65%<br>Range 1 Critical+</div>
      </div>
    </div>
    <div class="panel">
      <h3>Combat log</h3>
      <div id="log"></div>
    </div>
  </div>
</div>
<div id="phase">Turn 1 — Player turn</div>
<div id="bottom">
  <button id="btn-end" onclick="endPlayerTurn()">Turn End</button>
  <button id="btn-wait" onclick="waitUnit()" disabled>Wait</button>
  <button id="btn-restart" class="danger" onclick="initGame()">Restart</button>
</div>
</div>
<script>
const COLS=10,ROWS=10,TS=40;
```

```
const cv=document.getElementById('c');
cv.width=COLS*TS;cv.height=ROWS*TS;
const ctx=cv.getContext('2d');

const WEAPONS={
  sword:{name:'Sword',icon:'□□',atk:15,hit:90,range:1,color:'#4fc3f7'},
  bow:  {name:'Bow',icon:'□□',atk:10,hit:80,range:2,color:'#a5d6a7'},
  spear:{name:'Spear',icon:'□□',atk:13,hit:85,range:1,color:'#ce93d8',ctrAtk:5},
  axe:  {name:'Axe',icon:'□□',atk:20,hit:65,range:1,color:'#ffab91'},
};

const TERRAIN={
  plain:{name:'Yard',color:'#2d4a2d',def:0,move:1},
  forest:{name:'Bush',color:'#1a3a1a',def:2,move:2},
  hill:{name:'Hall',color:'#4a3a2a',def:1,move:2},
  water:{name:'Water',color:'#1a2a4a',def:0,move:999},
};

let MAP=[],units=[],sel=null,phase='player',turn=1,moveHighlight=[],attackHighlight=[];

function mkMap(){
  MAP=[];
  const layout=[
    'pppppppppf',
    'ppfppphhpp',
    'pffpppphpp',
    'ppppwwpppp',
    'ppphwwphpp',
    'pppphhhppp',
    'ppfpppppfp',
    'pppppppppf',
    'pfpphhpppf',
    'pppppppppp',
  ];
  const keys={p:'plain',f:'forest',h:'hill',w:'water'};
  for(let r=0;r<ROWS;r++){MAP[r]=[];for(let
c=0;c<COLS;c++)MAP[r][c]=keys[layout[r][c]]||'plain';}
}

function mkUnits(){
```

```
  units=[
    {id:0,name:'Sword
man',team:'player',weapon:'sword',hp:30,maxHp:30,atk:0,def:3,spd:5,x:0,y:8,moved:false,acted:f
alse,color:'#4fc3f7'},
    {id:1,name:'Spear
man',team:'player',weapon:'spear',hp:28,maxHp:28,atk:0,def:2,spd:4,x:1,y:9,moved:false,acted:f
alse,color:'#ce93d8'},

{id:2,name:'Archer',team:'player',weapon:'bow',hp:22,maxHp:22,atk:0,def:1,spd:6,x:0,y:7,moved:
false,acted:false,color:'#a5d6a7'},
    {id:3,name:'Axe
man',team:'player',weapon:'axe',hp:35,maxHp:35,atk:0,def:4,spd:3,x:2,y:9,moved:false,acted:fal
se,color:'#ffab91'},
    {id:4,name:'Sword
goblin',team:'enemy',weapon:'sword',hp:20,maxHp:20,atk:0,def:1,spd:4,x:9,y:1,moved:false,acted
:false,color:'#ef5350'},
    {id:5,name:'Axe
goblin',team:'enemy',weapon:'axe',hp:22,maxHp:22,atk:0,def:1,spd:3,x:8,y:0,moved:false,acted:f
alse,color:'#ef5350'},
    {id:6,name:'Spear
goblin',team:'enemy',weapon:'spear',hp:18,maxHp:18,atk:0,def:2,spd:5,x:9,y:2,moved:false,acted
:false,color:'#ff7043'},
    {id:7,name:'Bow
goblin',team:'enemy',weapon:'bow',hp:16,maxHp:16,atk:0,def:0,spd:6,x:7,y:0,moved:false,acted:f
alse,color:'#ef5350'},
  ];
}

function initGame(){
  mkMap();mkUnits();sel=null;phase='player';turn=1;moveHighlight=[];attackHighlight=[];
  document.getElementById('log').innerHTML='';
  setPhaseText();
  render();
  document.getElementById('btn-end').disabled=false;
}

function getUnit(x,y){return units.find(u=>u.x===x&&u.y===y&&u.hp>0);}

function inBounds(x,y){return x>=0&&y>=0&&x<COLS&&y<ROWS;}
```

```
function terrainMoveCost(x,y){return TERRAIN[MAP[y][x]].move;}

function getMoveCells(unit){
  const moveRange=3;
  const visited={};
  const q=[{x:unit.x,y:unit.y,cost:0}];
  visited[`${unit.x},${unit.y}`]=0;
  const cells=[];
  while(q.length){
    const cur=q.shift();
    cells.push({x:cur.x,y:cur.y});
    const dirs=[[1,0],[-1,0],[0,1],[0,-1]];
    for(const [dx,dy] of dirs){
      const nx=cur.x+dx,ny=cur.y+dy;
      const key=`${nx},${ny}`;
      if(!inBounds(nx,ny))continue;
      const cost=cur.cost+terrainMoveCost(nx,ny);
      if(cost>moveRange)continue;
      const occ=getUnit(nx,ny);
      if(occ&&occ.team!==unit.team)continue;

if(visited[key]===undefined||visited[key]>cost){visited[key]=cost;q.push({x:nx,y:ny,cost});}
    }
  }
  return cells.filter(c=>!(c.x===unit.x&&c.y===unit.y));
}

function getAttackCells(unit,fromX,fromY){
  const wep=WEAPONS[unit.weapon];
  const cells=[];
  for(let r=0;r<ROWS;r++)for(let c=0;c<COLS;c++){
    const dist=Math.abs(c-fromX)+Math.abs(r-fromY);
    if(dist>=1&&dist<=wep.range)cells.push({x:c,y:r});
  }
  return cells;
}

function dist(a,b){return Math.abs(a.x-b.x)+Math.abs(a.y-b.y);}

function calcDmg(attacker,defender){
```

```
  const wep=WEAPONS[attacker.weapon];
  const hit=Math.random()*100<wep.hit;
  if(!hit)return{dmg:0,hit:false};
  const dmg=Math.max(1,wep.atk+attacker.atk-
defender.def+TERRAIN[MAP[defender.y][defender.x]].def*-1);
  return{dmg,hit:true};
}

function doAttack(attacker,defender){
  const awep=WEAPONS[attacker.weapon];
  const dwep=WEAPONS[defender.weapon];
  const {dmg,hit}=calcDmg(attacker,defender);
  if(hit){defender.hp=Math.max(0,defender.hp-dmg);addLog(`${attacker.name}→${defender.name}
${dmg} Damage!`);}
  else addLog(`${attacker.name}→${defender.name} Miss!`);

  if(defender.hp>0){
    const defRange=dwep.range;
    const d=dist(attacker,defender);
    if(d<=defRange){
      const bonus=(defender.weapon==='spear'?dwep.ctrAtk||0:0);
      const r2=calcDmg({...defender,atk:defender.atk+bonus},attacker);
      if(r2.hit){attacker.hp=Math.max(0,attacker.hp-r2.dmg);addLog(`${defender.name}'s
Counter! ${r2.dmg} Damage!`);}
      else addLog(`${defender.name} Counter - Miss!`);
    }
  }
  if(defender.hp<=0)addLog(`☠ ${defender.name} was defeat!`);
  if(attacker.hp<=0)addLog(`☠ ${attacker.name} was defeat!`);
  checkWin();
}

function addLog(msg){
  const el=document.getElementById('log');
  const d=document.createElement('div');d.textContent=msg;
  el.appendChild(d);el.scrollTop=el.scrollHeight;
}

function checkWin(){
  const pAlive=units.filter(u=>u.team==='player'&&u.hp>0);
```

```
  const eAlive=units.filter(u=>u.team==='enemy'&&u.hp>0);
  if(pAlive.length===0){setTimeout(()=>{addLog('□□Enemies are victorious!');},100);}
  else if(eAlive.length===0){setTimeout(()=>{addLog('□□Player is victorious!');},100);}
}

function setPhaseText(){
  document.getElementById('phase').textContent=`Turn ${turn} — ${phase==='player'?'Player
turn':'Enemy turn'}`;
}

// Rendering
const COLORS={
  moveable:'rgba(80,180,255,0.25)',
  attackable:'rgba(255,80,80,0.28)',
  selected:'rgba(255,220,80,0.35)',
};

function render(){
  ctx.clearRect(0,0,cv.width,cv.height);
  // terrain
  for(let r=0;r<ROWS;r++)for(let c=0;c<COLS;c++){
    ctx.fillStyle=TERRAIN[MAP[r][c]].color;
    ctx.fillRect(c*TS,r*TS,TS,TS);
    ctx.strokeStyle='rgba(0,0,0,0.3)';ctx.lineWidth=0.5;
    ctx.strokeRect(c*TS,r*TS,TS,TS);
  }
  // highlights
  for(const h of moveHighlight){
    ctx.fillStyle=COLORS.moveable;ctx.fillRect(h.x*TS,h.y*TS,TS,TS);
  }
  for(const h of attackHighlight){
    ctx.fillStyle=COLORS.attackable;ctx.fillRect(h.x*TS,h.y*TS,TS,TS);
  }
  if(sel){
    ctx.fillStyle=COLORS.selected;ctx.fillRect(sel.x*TS,sel.y*TS,TS,TS);
  }
  // units
  for(const u of units){
    if(u.hp<=0)continue;
    const px=u.x*TS,py=u.y*TS;
```

```
    ctx.fillStyle=u.team==='player'?(u.moved?'#555':u.color):(u.color);
    ctx.beginPath();ctx.roundRect(px+5,py+5,TS-10,TS-10,4);ctx.fill();
    if(u===sel){ctx.strokeStyle='#ffe066';ctx.lineWidth=2;ctx.stroke();}
    const wep=WEAPONS[u.weapon];
    ctx.font='16px serif';ctx.textAlign='center';ctx.textBaseline='middle';
    ctx.fillText(wep.icon,px+TS/2,py+TS/2);
    // hp bar
    const barW=TS-8,barH=4,bx=px+4,by=py+TS-8;
    ctx.fillStyle='#222';ctx.fillRect(bx,by,barW,barH);
    ctx.fillStyle=u.team==='player'?'#4caf50':'#f44336';
    ctx.fillRect(bx,by,barW*(u.hp/u.maxHp),barH);
    // team indicator
    ctx.fillStyle=u.team==='player'?'#4fc3f7':'#ef5350';
    ctx.fillRect(px+4,py+4,8,3);
  }
}

// State machine
let state='idle'; // idle | selected | moved
let selMoves=[];
let selAttacks=[];
let movedPos=null;

cv.addEventListener('click',e=>{
  const rect=cv.getBoundingClientRect();
  const scaleX=cv.width/rect.width,scaleY=cv.height/rect.height;
  const mx=Math.floor((e.clientX-rect.left)*scaleX/TS);
  const my=Math.floor((e.clientY-rect.top)*scaleY/TS);
  if(!inBounds(mx,my))return;
  handleClick(mx,my);
});

function handleClick(cx,cy){
  if(phase!=='player')return;
  const clicked=getUnit(cx,cy);

  if(state==='idle'){
    if(clicked&&clicked.team==='player'&&!clicked.acted){
      sel=clicked;
      selMoves=getMoveCells(clicked);
```

```
      selAttacks=getAttackCells(clicked,clicked.x,clicked.y);
      moveHighlight=selMoves;
      attackHighlight=selAttacks.filter(a=>{const e=getUnit(a.x,a.y);return
e&&e.team==='enemy';});
      state='selected';
      showUnitInfo(clicked);
      document.getElementById('btn-wait').disabled=false;
    }else{
      clearSel();
    }
  } else if(state==='selected'){
    // Click same unit = deselect
    if(clicked===sel){clearSel();return;}
    // Click enemy in range = attack
    if(clicked&&clicked.team==='enemy'){
      const inRange=selAttacks.some(a=>a.x===cx&&a.y===cy);
      if(inRange){
        doAttack(sel,clicked);
        sel.acted=true;sel.moved=true;
        clearSel();render();return;
      }
    }
    // Click move cell
    if(selMoves.some(m=>m.x===cx&&m.y===cy)&&!getUnit(cx,cy)){
      movedPos={x:sel.x,y:sel.y};
      sel.x=cx;sel.y=cy;
      sel.moved=true;
      // Show attack options from new pos
      selAttacks=getAttackCells(sel,cx,cy);
      const enemies=selAttacks.filter(a=>{const e=getUnit(a.x,a.y);return
e&&e.team==='enemy';});
      moveHighlight=[];
      attackHighlight=enemies;
      state='moved';
      render();return;
    }
    // Click different ally
    if(clicked&&clicked.team==='player'&&!clicked.acted){
      sel=clicked;
      selMoves=getMoveCells(clicked);
```

```
      selAttacks=getAttackCells(clicked,clicked.x,clicked.y);
      moveHighlight=selMoves;
      attackHighlight=selAttacks.filter(a=>{const e=getUnit(a.x,a.y);return
e&&e.team==='enemy';});
      showUnitInfo(clicked);
      render();return;
    }
    clearSel();
  } else if(state==='moved'){
    // Click enemy to attack
    if(clicked&&clicked.team==='enemy'){
      const inRange=selAttacks.some(a=>a.x===cx&&a.y===cy);
      if(inRange){
        doAttack(sel,clicked);
        sel.acted=true;
        clearSel();render();return;
      }
    }
    // Click empty = undo move
    if(!clicked){
      if(movedPos){sel.x=movedPos.x;sel.y=movedPos.y;sel.moved=false;}
      selMoves=getMoveCells(sel);
      selAttacks=getAttackCells(sel,sel.x,sel.y);
      moveHighlight=selMoves;
      attackHighlight=selAttacks.filter(a=>{const e=getUnit(a.x,a.y);return
e&&e.team==='enemy';});
      state='selected';movedPos=null;render();return;
    }
  }
  render();
}

function clearSel(){
  sel=null;state='idle';moveHighlight=[];attackHighlight=[];movedPos=null;
  document.getElementById('btn-wait').disabled=true;
  document.getElementById('unit-name').textContent='Select Unit';
  document.getElementById('unit-name').style.color='#aaa';
  document.getElementById('unit-stats').innerHTML='';
}
```

```
function waitUnit(){
  if(!sel)return;
  sel.moved=true;sel.acted=true;
  clearSel();render();
}

function showUnitInfo(u){
  const wep=WEAPONS[u.weapon];
  document.getElementById('unit-name').textContent=`${wep.icon} ${u.name}`;
  document.getElementById('unit-name').style.color=u.color;
  document.getElementById('unit-stats').innerHTML=`
    <div class="bar-row"><span class="bar-label">HP</span><div class="bar-bg"><div class="bar-
fill hp-bar" style="width:${(u.hp/u.maxHp)*100}%"></div></div><span style="font-
size:11px;margin-left:4px">${u.hp}/${u.maxHp}</span></div>
    <div style="font-size:11px;color:#aaa;margin-top:4px">
      Weapon: ${wep.name} | Atk: ${wep.atk} | Acc: ${wep.hit}%<br>
      Range: ${wep.range} | Def: ${u.def} | Spd: ${u.spd}
    </div>
    <div style="font-size:11px;color:${u.moved?'#f66':'#4fc'};margin-
top:3px">${u.acted?'Acted':u.moved?'Moved':'Movable'}</div>
  `;
}

function endPlayerTurn(){
  clearSel();
  phase='enemy';
  setPhaseText();
  document.getElementById('btn-end').disabled=true;
  render();
  setTimeout(enemyTurn,600);
}

function enemyTurn(){
  const enemies=units.filter(u=>u.team==='enemy'&&u.hp>0);
  const players=units.filter(u=>u.team==='player'&&u.hp>0);
  let i=0;
  function doNext(){
    if(i>=enemies.length){
      // Reset
      units.forEach(u=>{u.moved=false;u.acted=false;});
```

```
      phase='player';turn++;
      setPhaseText();
      document.getElementById('btn-end').disabled=false;
      render();return;
    }
    const enemy=enemies[i];i++;
    if(enemy.hp<=0){doNext();return;}
    // Find nearest player
    const pAlive=units.filter(u=>u.team==='player'&&u.hp>0);
    if(!pAlive.length){render();return;}
    const target=pAlive.reduce((a,b)=>dist(enemy,a)<dist(enemy,b)?a:b);
    const wep=WEAPONS[enemy.weapon];
    const d=dist(enemy,target);
    if(d<=wep.range){
      // Attack
      doAttack(enemy,target);
    } else {
      // Move toward target
      const moves=getMoveCells(enemy);
      // pick cell closest to target
      let best=null,bestD=999;
      for(const m of moves){
        const md=Math.abs(m.x-target.x)+Math.abs(m.y-target.y);
        if(md<bestD&&!getUnit(m.x,m.y)){bestD=md;best=m;}
      }
      if(best){enemy.x=best.x;enemy.y=best.y;}
      // Try attack from new pos
      const atks=getAttackCells(enemy,enemy.x,enemy.y);
      const inRange=atks.find(a=>a.x===target.x&&a.y===target.y);
      if(inRange)doAttack(enemy,target);
    }
    enemy.moved=true;enemy.acted=true;
    render();
    setTimeout(doNext,500);
  }
  doNext();
}

initGame();
</script>
```

```
</body>
</html>
```