

```

import React, { useState, useEffect } from 'react';
import { BrickWall, User, Droplet, Zap, Heart, Wrench, Package, Cpu,
Bug, AlertTriangle, CloudRain, Thermometer, Smile, Frown } from
'lucide-react';

// --- M3 核心數據結構模擬 (Structural Certainty) ---

/**
 * @typedef {object} BoundaryData
 * @property {string} materialId - 邊界材料 ID
 * @property {number} structuralHealth - 結構健康度 (0.0 - 1.0)
 * @property {boolean} isBlocked - 是否完全阻塞通行
 */

/**
 * @typedef {object} GridCellData
 * @property {string} id - 唯一 ID, 即 "x_y" 座標
 * @property {string | null} entityId - 單元格內的主要實體 ID
 * @property {number} supportCapacity - 總承重能力 (kg)
 * @property {number} loadTotal - 當前承載的總載荷 (kg)
 * @property {number} temperature - 溫度 (K)
 * @property {{ [key: string]: number }} contaminants - 汚染物與微生物
(使用 Object 確保穩定性)
 * @property {{ [key: string]: BoundaryData }} boundaries - 邊界數據
 */

/**
 * @typedef {object} NPCState
 * @property {string} id - 唯一 ID
 * @property {'StarGod' | 'Kancolle' | 'Catgirl' | 'Human'} archetype
- 角色原型
 * @property {string} factionId - 陣營 ID
 * @property {number} health - 生命值
 * @property {number} hydration - 水合度
 * @property {number} immunityIndex - 免疫力
 * @property {boolean} isSick - 是否生病
 * @property {number} diseaseProgress - 疾病進程率
 * @property {number} emotionAnger - 憤怒值
 * @property {number} emotionFear - 恐懼值
 * @property {number} emotionJoy - 快樂值
 * @property {{ [key: string]: number }} affectionGraph - 社交關係圖 (使
用 Object 確保穩定性)
 * @property {string[]} socialTags - 社交標籤
 */
*/

* @typedef {object} CraftingRecipe
* @property {string} recipeId - 配方 ID

```

```

* @property {string} outputItemId - 輸出產品 ID
* @property {number} minSkillLevel - 最低技能等級
* @property {{ materialId: string, quantity: number }[]} inputMaterials - 輸入材料列表
* @property {string} outputFormulaId - M3 確定性公式 ID
*/

```

// --- 模擬數據生成 (Dummy Data) ---

```

const MOCK_NPC_ID = "NPC_Kancolle_001";
const MOCK_GRID_ID = "5_5";

/** @type {NPCState} */
const initialNPCState = {
  id: MOCK_NPC_ID,
  archetype: 'Kancolle',
  factionId: 'Miya_Star_Gods',
  health: 0.85,
  hydration: 0.92,
  immunityIndex: 0.75,
  isSick: true,
  diseaseProgress: 0.15,
  emotionAnger: 0.1,
  emotionFear: 0.45,
  emotionJoy: 0.8,
  affectionGraph: {
    'Player_001': 0.65,
    'NPC_Catgirl_002': -0.3,
  },
  socialTags: ['Commander', 'Engineer', 'Loyalist'],
};

/** 
 * @param {string} id
 * @param {string | null} entityId
 * @returns {GridCellData}
 */
const createMockCellData = (id, entityId) => ({
  id,
  entityId,
  supportCapacity: 50000,
  loadTotal: Math.random() * 50000,
  temperature: 295.15, // 22 C
  contaminants: entityId === 'Furnace' ? {'Carbon_Monoxide': 0.02} : {},
  boundaries: {
    x_pos: { materialId: 'Steel', structuralHealth: 1.0,

```

```

        isBlocked: true },
            x_neg: { materialId: 'Wood', structuralHealth: 0.5, isBlocked:
true },
            y_pos: { materialId: 'Air', structuralHealth: 1.0, isBlocked:
false },
            y_neg: { materialId: 'Steel', structuralHealth: 1.0,
isBlocked: true },
            z_pos: { materialId: 'Steel', structuralHealth: 1.0,
isBlocked: true },
            z_neg: { materialId: 'Steel', structuralHealth: 1.0,
isBlocked: true },
        }
    );
}

const mockRecipes = [
    { recipeId: 'R-001', outputItemId: 'High_Carbon_Steel',
minSkillLevel: 5, inputMaterials: [{ materialId: 'Iron_Ore', quantity:
10 }, { materialId: 'Coal', quantity: 5 }], outputFormulaId:
'METALLURGY_HARDNESS' },
    { recipeId: 'R-002', outputItemId: 'Anti-Toxin_Serum',
minSkillLevel: 8, inputMaterials: [{ materialId: 'Water', quantity: 3
}, { materialId: 'Herb_X', quantity: 1 }], outputFormulaId:
'CHEMICAL_POTENCY' },
];

```

// 網格大小

```

const GRID_SIZE = 10;

```

// --- 輔助組件 ---

```

const IconMap = {
    Wall: BrickWall,
    NPC: User,
    Water: Droplet,
    Furnace: Zap,
    Workbench: Wrench,
    default: Package,
};

```

```

/**
 * M3 狀態進度條
 * @param {object} props
 * @param {string} props.label
 * @param {number} props.value - 0.0 to 1.0
 * @param {string} props.colorClass - Tailwind color class
 */
const ProgressBar = ({ label, value, colorClass }) => (
    <div className="mb-2">

```

```

        <div className="flex justify-between text-xs mb-1">
            <span className="font-mono">{label}</span>
            <span className="font-semibold">{(value *
100).toFixed(1)}%</span>
        </div>
        <div className="w-full bg-gray-700 rounded-full h-2.5">
            <div className={`${`h-2.5 rounded-full ${colorClass}`}`}
style={{ width: `${value * 100}%` }}></div>
        </div>
    </div>
);

const DataSection = ({ title, children }) => (
    <div className="border border-gray-600 rounded-lg p-3 mb-4
bg-gray-800/50">
        <h3 className="text-sm font-bold text-teal-400 mb-2 border-b
border-gray-700 pb-1">{title}</h3>
        {children}
    </div>
);

const M3ValueDisplay = ({ label, value, unit }) => (
    <div className="flex justify-between text-xs border-b
border-gray-800 py-1">
        <span className="text-gray-400">{label}</span>
        <span className="font-mono text-white">{value.toFixed(2)}
        <span className="text-gray-500">{unit}</span></span>
    </div>
);

// --- 核心組件 ---

/***
 * @param {object} props
 * @param {GridCellData} props.cell
 * @param {boolean} props.isSelected
 * @param {function(): void} props.onClick
 * @param {boolean} props.npcPresent
 */
const GridCell = ({ cell, isSelected, onClick, npcPresent }) => {
    const IconComponent = IconMap[cell.entityId] || IconMap.default;
    const loadRatio = cell.loadTotal / cell.supportCapacity;

    let bgColor = 'bg-gray-900/50';
    let borderColor = 'border-gray-700';

    if (loadRatio > 0.8) bgColor = 'bg-red-900/40'; // 應力過高
    else if (loadRatio > 0.6) bgColor = 'bg-yellow-900/40'; // 應力警告

```

```

// 強制選中時的背景和邊框
if (isSelected) {
    borderColor = 'border-teal-400 border-2';
    bgColor = 'bg-teal-900/50';
}

return (
    <div
        className={`${`w-full h-full flex items-center justify-center
p-1 cursor-pointer transition duration-150 ease-in-out ${bgColor}
${borderColor} hover:bg-gray-700/50 relative`}
        onClick={onClick}
    >
        {npcPresent && (
            <User className="w-4 h-4 text-pink-400 absolute
top-0.5 right-0.5" strokeWidth={2.5} />
        )}
        <IconComponent className="w-6 h-6 text-gray-400" />
        <span className="text-xs font-mono absolute bottom-0.5
left-0.5 text-gray-500">{cell.id}</span>
    </div>
);
};

/***
 * 數據檢查器，顯示單元格或 NPC 的 M3 確定性狀態
 * @param {object} props
 * @param {GridCellData | NPCState | null} props.selectedEntity
 * @param {'cell' | 'npc' | 'none'} props.type
 * @param {function(): void} props.onCraft
 */
const DataInspector = ({ selectedEntity, type, onCraft }) => {
    if (!selectedEntity) {
        return (
            <div className="p-4 text-center text-gray-500">
                <p className="text-lg font-mono">--- M3 邏輯核心 ---</p>
                <p className="mt-2">請在網格中點擊一個單元格或實體來檢查 M3
確定性狀態。</p>
            </div>
        );
    }

    if (type === 'npc') {
        /** @type {NPCState} */
        const npc = selectedEntity;
        const healthColor = npc.health > 0.5 ? 'bg-green-500' :
        npc.health > 0.2 ? 'bg-yellow-500' : 'bg-red-500';
    }
}

```

```

        return (
            <div className="p-4 overflow-y-auto max-h-[90vh] text-sm">
                <h2 className="text-2xl font-bold text-pink-400
border-b border-pink-700 pb-2 mb-4 flex items-center">
                    <User className="mr-2" /> {npc.id}
                ({npc.archetype})
                </h2>
                <DataSection title="I. M3 生理與健康狀態 (Determinism)">
                    <ProgressBar label="生命值 (Health)" value={npc.health} colorClass={healthColor} />
                    <ProgressBar label="水合度 (Hydration)" value={npc.hydration} colorClass="bg-blue-500" />
                    <ProgressBar label="免疫力 (Immune Index)" value={npc. immunityIndex} colorClass="bg-purple-500" />
                    <M3ValueDisplay label="疾病進程率 (P_Sick)" value={npc.diseaseProgress * 10} unit="/hr" />
                    <div className={`${`mt-2 p-2 rounded-md ${npc.isSick ? 'bg-red-900/50 text-red-300' : 'bg-green-900/50 text-green-300'}`}>
                        {npc.isSick ? <><Bug className="inline w-4 h-4 mr-1" /> 狀態：生病中 (高壓力因子)</> : '狀態：健康穩定'}
                    </div>
                </DataSection>

                <DataSection title="II. 情感與行為量化 (Quantified Emotion)">
                    <div className="flex justify-around text-center mb-3">
                        <div className="flex flex-col items-center">
                            <Smile className={`${`w-6 h-6 ${npc.emotionJoy > 0.7 ? 'text-yellow-400' : 'text-gray-500'}`}`} />
                            <span className="text-xs mt-1">快樂：{npc.emotionJoy.toFixed(2)}</span>
                        </div>
                        <div className="flex flex-col items-center">
                            <Frown className={`${`w-6 h-6 ${npc.emotionAnger > 0.5 ? 'text-red-400' : 'text-gray-500'}`}`} />
                            <span className="text-xs mt-1">憤怒：{npc.emotionAnger.toFixed(2)}</span>
                        </div>
                        <div className="flex flex-col items-center">
                            <AlertTriangle className={`${`w-6 h-6 ${npc.emotionFear > 0.5 ? 'text-orange-400' : 'text-gray-500'}`}`} />
                            <span className="text-xs mt-1">恐懼：{npc.emotionFear.toFixed(2)}</span>
                        </div>
                    </div>
                    <M3ValueDisplay label="家庭關係權重" value={1.0} />
                </DataSection>
            </div>
        )
    )

```

```

unit=" (Max Priority) " />
                <p className="text-xs mt-2 text-gray-400">社交標籤:<br/>
{npc.socialTags.join(', ')}</p>
            </DataSection>

            <DataSection title="III. 社交拓撲 (Affection Graph)">

{Object.entries(npc.affectionGraph).map(([targetId, value]) => (
                <div key={targetId} className="flex justify-between text-xs py-1 border-b border-gray-700 last:border-b-0">
                    <span
                        className="text-gray-300">{targetId}</span>
                    <span className={`${fontSemibold ${value > 0 ? 'text-green-400' : 'text-red-400'}}`}>{value.toFixed(2)}</span>
                </div>
            ))}
            </DataSection>
        </div>
    );
}

// Grid Cell Data View (CDDA Complexity)
/** @type {GridCellData} */
const cell = selectedEntity;
const loadRatio = cell.loadTotal / cell.supportCapacity;
const loadColor = loadRatio > 0.8 ? 'bg-red-500' : loadRatio > 0.6
? 'bg-yellow-500' : 'bg-green-500';

return (
    <div className="p-4 overflow-y-auto max-h-[90vh] text-sm">
        /* 已移除 LaTeX 符號，確保 JSX 解析穩定性 */
        <h2 className="text-2xl font-bold text-teal-400 border-b border-teal-700 pb-2 mb-4 flex items-center">
            <Cpu className="mr-2" /> 1立方米 單元格: {cell.id}
        </h2>

        <DataSection title="I. M3 結構確定性 (Structural Certainty)">
            <ProgressBar label="結構健康度 (Boundary Avg)" value={Object.values(cell.boundaries).reduce((acc, b) => acc + b.structuralHealth, 0) / 6} colorClass="bg-blue-500" />
            <ProgressBar label="應力載荷 (Load Ratio)" value={loadRatio} colorClass={loadColor} />
            <M3ValueDisplay label="總承重能力 (Capacity)" value={cell.supportCapacity} unit="kg" />
            <M3ValueDisplay label="當前總載荷 (Load Total)" value={cell.loadTotal} unit="kg" />
        </DataSection>
    </div>
)

```

```

        </DataSection>

        <DataSection title="II. M3 環境與化學場 (CDDA Complexity)">
            <M3ValueDisplay label="溫度 (Temperature)" value={cell.temperature} unit="K" />
            <M3ValueDisplay label="水體積 (Water Volume)" value={cell.boundaries.y_pos.structuralHealth} unit="m³" />
            <h4 className="mt-3 text-xs font-semibold text-gray-400">污染物 / 微生物載荷 (Contaminants):</h4>
            {Object.entries(cell.contaminants).map(([name, concentration]) => (
                <M3ValueDisplay key={name} label={name} value={concentration * 100} unit="%" />
            ))}
        </DataSection>

        <DataSection title="III. 邊界阻塞索引 (BBI Status)">
            {Object.entries(cell.boundaries).map(([dir, boundary]) => (
                <div key={dir} className="flex justify-between text-xs py-1 border-b border-gray-800">
                    <span className="text-gray-400 font-mono">{dir}</span>
                    <span className="text-white">
                        {boundary.materialId}
                        <span className="ml-2 font-semibold ${boundary.isBlocked ? 'text-red-500' : 'text-green-500'}">
                            ({boundary.structuralHealth.toFixed(2)})
                        </span>
                    </span>
                </div>
            )));
        <div>
            {cell.entityId === 'Furnace' && (
                <button onClick={onCraft} className="w-full mt-3 p-2 bg-indigo-600 hover:bg-indigo-700 rounded-md transition duration-150 text-white font-semibold flex items-center justify-center">
                    <Wrench className="w-4 h-4 mr-2" /> 進入模塊化製作
                    </button>
            )}
        </DataSection>
    </div>
);
};

```

```

/**
 * 模塊化製作系統 (M3 確定性邏輯模擬)
 * @param {object} props
 * @param {function(): void} props.onClose
 */
const CraftingModal = ({ onClose }) => {
    const [selectedRecipe, setSelectedRecipe] =
useState(mockRecipes[0]);
    const [status, setStatus] = useState('idle'); // 'idle' |
'calculating' | 'success' | 'failure'
    const [resultMsg, setResultMsg] = useState('');

    const handleCraft = () => {
        if (!selectedRecipe) return;

        setStatus('calculating');
        setResultMsg('');

        // 模擬 M3 邏輯核心計算延遲 (計算真率)
        setTimeout(() => {
            // M3 確定性邏輯模擬 :
            const playerSkill = 6;
            if (playerSkill < selectedRecipe.minSkillLevel) {
                setStatus('failure');
                setResultMsg(`[M3 FAIL]: 技能等級 ${playerSkill} 低於要
求 ${selectedRecipe.minSkillLevel})。確定性輸出：低品質廢料。`);
            } else {
                // 模擬確定性結果
                const baseQuality = 0.85;
                const finalQuality = baseQuality * (1 + (playerSkill -
selectedRecipe.minSkillLevel) * 0.05);
                setStatus('success');
                setResultMsg(`[M3 SUCCESS]: 確定性輸出
[${selectedRecipe.outputItemId}]。品質屬性 (硬度/純度)：
${finalQuality.toFixed(3)} (M3 公式計算)`);
            }
        }, 1500); // 模擬 1.5 秒的計算真率延遲
    };

    return (
        <div className="fixed inset-0 bg-black bg-opacity-70 flex
items-center justify-center z-50">
            <div className="bg-gray-900 border border-teal-600 p-6
rounded-xl w-11/12 max-w-lg shadow-2xl">
                <h2 className="text-2xl font-bold text-teal-400 mb-4
border-b border-gray-700 pb-2">模塊化製作系統 (M3 確定性)</h2>
                <DataSection title="I. 配方選擇與輸入">

```

```

        <label className="block text-sm font-medium
text-gray-400 mb-2">選擇配方:</label>
        <select
            className="w-full bg-gray-700 border
border-gray-600 rounded-md p-2 text-white"
            onChange={(e) =>
setSelectedRecipe(mockRecipes.find(r => r.recipeId === e.target.value)
|| null)}
        >
            {mockRecipes.map(r => (
                <option key={r.recipeId}
value={r.recipeId}>
                    {r.outputItemId} ({r.outputFormulaId})
                </option>
            )))
        </select>

        {selectedRecipe && (
            <div className="mt-4 text-xs text-gray-300">
                <p className="font-semibold text-teal-300
mb-1">輸入需求:</p>
                {selectedRecipe.inputMaterials.map((mat,
index) => (
                    <p key={index} className="ml-2">-
{mat.materialId} x {mat.quantity}</p>
                )))
            <p className="mt-2 text-gray-500">最低技能等
級: {selectedRecipe.minSkillLevel} | 核心公式:
{selectedRecipe.outputFormulaId}</p>
            </div>
        )}
    </DataSection>

    <div className="mt-6 flex justify-between
items-center">
        <button
            onClick={handleCraft}
            disabled={status === 'calculating'}
            className="px-6 py-2 bg-green-600
hover:bg-green-700 rounded-lg text-white font-bold transition
duration-150 disabled:bg-gray-500 flex items-center"
        >
            {status === 'calculating' ? (
                <div className="animate-spin mr-2"><Cpu
className="w-4 h-4" /></div>
            ) : (
                <Wrench className="w-4 h-4 mr-2" />
            )}
        </button>
    </div>

```

```

        {status === 'calculating' ? 'M3 邏輯計算中...' :
'點擊製造 (確定性結果)'}
            </button>
            <button onClick={onClose} className="px-4 py-2
bg-gray-600 hover:bg-gray-700 rounded-lg text-white">
                關閉
            </button>
        </div>

        {resultMsg && (
            <div className={`${`mt-4 p-3 rounded-lg text-sm
font-mono ${status === 'success' ? 'bg-green-900/50 text-green-300' :
'bg-red-900/50 text-red-300'}`}>
                {resultMsg}
            </div>
        )}
    </div>
</div>
);
};

const App = () => {
    const [gridData, setGridData] = useState([]);
    const [selectedEntity, setSelectedEntity] = useState(null);
    const [selectedType, setSelectedType] = useState('none'); // 
'cell' | 'npc' | 'none'
    const [isCraftingModalOpen, setIsCraftingModalOpen] =
useState(false);

    // 1. 初始化網格數據
    useEffect(() => {
        const data = [];
        for (let y = 0; y < GRID_SIZE; y++) {
            for (let x = 0; x < GRID_SIZE; x++) {
                const id = `${x}_${y}`;
                let entityId = null;
                // 佈置實體
                if (x === 1 && y === 1) entityId = 'Wall';
                else if (x === 5 && y === 5) entityId = 'Furnace'; // 
MOCK_GRID_ID
                else if (x === 8 && y === 2) entityId = 'Workbench';
                data.push(createMockCellData(id, entityId));
            }
        }
        setGridData(data);
    }, []);

    // 2. 處理實體點擊選擇

```

```

const handleCellClick = (cell) => {
    // 模擬 NPC 邏輯：NPC 位於 MOCK_GRID_ID (5_5)
    if (cell.id === MOCK_GRID_ID) {
        setSelectedEntity(initialNPCState);
        setSelectedType('npc');
    } else {
        setSelectedEntity(cell);
        setSelectedType('cell');
    }
};

return (
    <div className="min-h-screen bg-gray-950 text-white font-sans p-2 lg:p-4">
        <header className="text-center mb-4 border-b border-gray-700 pb-2">
            <h1 className="text-3xl font-extrabold text-teal-400">ASI 工程模擬平台 (MVP) </h1>
            <p className="text-sm text-gray-500 font-mono">M1 效率核心 / M3 確定性邏輯</p>
        </header>

        <main className="flex flex-col lg:flex-row gap-4">
            {/* 網格視圖 - RimWorld 風格 */}
            <div className="lg:w-2/3 w-full bg-gray-900 border border-gray-700 rounded-xl shadow-lg p-4">
                <h2 className="text-xl font-bold text-gray-300 mb-3 flex items-center"><CloudRain className="w-5 h-5 mr-2" /> 米亞星系：俯視圖 (10x10)</h2>
                <div
                    className="grid w-full aspect-square border-4 border-gray-800"
                    style={{ gridTemplateColumns: `repeat(${GRID_SIZE}, 1fr)`, gridTemplateRows: `repeat(${GRID_SIZE}, 1fr)` }}
                >
                    {gridData.map(cell => (
                        <GridCell
                            key={cell.id}
                            cell={cell}
                            isSelected={
                                // 判斷是否選中：如果是 cell 類型且 ID 匹配，或如果是 npc 類型且 ID 匹配
                                (selectedType === 'cell' && selectedEntity?.id === cell.id) ||
                                (selectedType === 'npc' && cell.id === MOCK_GRID_ID)
                            }
                        >
                    ))
                
```

```

        onClick={() => handleCellClick(cell)}
        npcPresent={cell.id === MOCK_GRID_ID}
      />
    )) }
  </div>
<div className="mt-3 text-sm text-gray-500">
  應力顏色提示：<span className="text-yellow-500">
  黃 (警告)</span>, <span className="text-red-500">紅 (崩塌風險 >
  80%)</span>
</div>
</div>

 {/* 數據檢查器 - CDDA 複雜度風格 */}
<div className="lg:w-1/3 w-full bg-gray-900 border border-gray-700 rounded-xl shadow-lg">
  <h2 className="text-xl font-bold text-gray-300 p-4 border-b border-gray-700 flex items-center"><Cpu className="w-5 h-5 mr-2" /> 實體數據檢查器</h2>
  <DataInspector
    selectedEntity={selectedEntity}
    type={selectedType}
    onCraft={() => setIsCraftingModalOpen(true)}
  />
</div>
</main>

{isCraftingModalOpen && <CraftingModal onClose={() => setIsCraftingModalOpen(false)} />}

</div>
);
};

export default App;

```