# AetherGenesis: 星神生態圈 - 開發規格與實作細節

文件編號: AG-SPEC-V1.0 目的: 提供將核心設計藍圖 (\text{V1.1}) 轉化為可實作 React/Firestore 原型的詳細工程規範。

## I. 介面與使用者體驗 (UI/UX) 規範

### A. 介面佈局與特點

| 區域 | 名稱 | 主要功能與特點 | 技術要求 |
|---|---|---|---|
| 中央區 | 六角形網格區 **(Hex Grid)** | 視覺化行星的結構確定性域。支援觸控/滑鼠點擊、拖曳。地塊在 M3 達到密度閾值時平滑編譯(擴展)。 | SVG 或 Canvas 實現，確保在移動端(手機)上能響應式縮放，避免卡頓。 |
| 左側欄 | 星神核心儀表板 | 顯示六個核心屬性(M1-M6 抽象術語)的當前數值、趨勢線和狀態指示燈。 | 必須使用 Tailwind 實現響應式佈局，並在 M6 數值臨界時提供強烈的視覺警告。 |
| 頂部欄 | 資源總覽 | 顯示最重要的三到四個 M/L 元素(如 \text{L}_{\text{A}} 活化靈)的淨收入狀態(綠色↑/紅色↓)。 | 需有工具提示(Tooltip)顯示詳細的資源數據(庫存、產出)。 |
| 底部彈窗 | 操作/警告區 | 用於地塊點擊後的操作選單(建造、派遣、轉化)以及 M6 檢查點機制觸發時的強制彈窗。 | 必須使用 React Portals 實現，以確保在遊戲邏輯暫停時，警告 UI 能置於頂層。 |

### B. M-值核心屬性顯示 (星神核心儀表板)

所有核心屬性 (M1-M6) 均顯示為 **0.00** 到 **1.00** 之間的百分比或等級。

| 內部 M-值 | UI 術語 | 狀態指示燈顏色/含義 |
|---|---|---|
| \text{M}_1 效率核心 | 開發效率 (0.00 - 1.00) | 綠色 (高於 0.6) / 黃色 (0.3 - 0.6) / 紅色 (低於 0.3，導致所有生產速度減半) |
| \text{M}_3 邏輯核心 | 結構精確度 (0.00 - 1.00) | 藍色 (高於 0.8) / 閃爍黃色 (低於 0.5，導致物質轉化損失增加 50%) |
| \text{M}_6 治理安全 | 治安與穩定度 (0.00 - 1.00) | 穩定藍(高)/ 閃爍紅(低於 0.65，M6 鎖定模式觸發) |

## II. 程式邏輯與計算細節 (Programming Logic &

# Calculation)

## A. 核心遊戲循環與數據結構

### 1. 遊戲循環 (\text{GameLoop})

遊戲邏輯將以 每秒一次 **(1 Hz)** 的時間步長 (Time Step) 運行。

1. 資源計算: 基於 M1 效率和造物產出，更新所有 M/L 元素的淨收入。
2. **M-**值更新: 根據造物、實體和種群的修正，重新計算 M1 到 M6 的當前值。
3. 事件檢查: 檢查 M6 治安與穩定度是否達到臨界值 (0.65) 或觸發隨機事件。
4. 數據持久化: 將關鍵狀態(M-值、資源庫存)非同步寫入 Firestore。

### 2. M-值數據結構 (React State / Firestore Document)

核心狀態將儲存在單一的 StarDeityCore 文件中。

```
// StarDeityCore.ts
interface StarDeityCore {
    // 每個 M-值都包含當前值、基礎值、和修正值
    M1_Efficiency: { value: number; base: number; modifier: number; };
// 開發效率
    M2_Autonomy: { value: number; base: number; modifier: number; };
// 生態自愈力
    M3_Precision: { value: number; base: number; modifier: number; };
// 結構精確度
    M4_Resilience: { value: number; base: number; modifier: number; };
// 系統韌性
    M5_Knowledge: { value: number; base: number; modifier: number; };
// 知識核心
    M6_Security: { value: number; base: number; modifier: number; };
// 治安與穩定度
    LastUpdated: number; // 時間戳，用於計算離線資源
}
```

## B. 關鍵計算公式

### 1. M-值更新公式 (每秒執行)

所有 M-值均由基礎值、造物加成、實體加成和種群懲罰決定。

$$ \text{M}_n \text{ Value} = \text{M}_n \text{ Base} + \sum (\text{Building M}_n \text{ Buff}) + \sum (\text{Entity M}_n \text{ Buff}) - \sum (\text{Population M}_n \text{ Cost}) $$

- 造物加成: 例如，一個 \text{ServerFarm} 增加 +0.05 的 \text{M}_3 (結構精確度)。
- 種群懲罰: 例如，過多的人口或異星種族會對 \text{M}_6 (治安與穩定度) 產生線性懲罰。

### 2. 資源轉化公式 (M3 核心)

物質 **(\text{M})** 轉化為靈 **(\text{L})** 的效率由 **\text{M}_3** 結構精確度決定。

- 備註：如果 \text{M}_3 為 1.00，轉化效率達到 2\times 基礎值；如果 \text{M}_3 為 0.00，轉化效率僅為 1\times 基礎值（即 50\% 的基礎產出）。

### 3. M6 檢查點機制 (M6 Locking Mode)

當玩家執行不可逆決策時觸發，程式執行以下邏輯：

1. 風險評估：計算 \text{V}_{\text{Total}} (不可逆決策的總風險值)。
2. 條件檢查：\text{IF } (\text{M}_6 \text{ Value} < 0.65 \text{ OR } \text{V}_{\text{Total}} \ge 0.65)
3. 觸發鎖定：
    - **UI** 反饋：彈出警告區，畫面暫停，顯示**「M6 治理核心已介入」**。
    - 修正行動：最終執行結果 (\text{ActualAction}) 嚴格按照以下公式修正：
        - \text{M}_{\text{MinPath}}：玩家原決策的邏輯向量（最小化損失）。
        - \text{M}_2 \text{ Value}：當前生態自愈力的值，決定了探索因子的強度。
        - \text{RandomFactor}：0.5 \sim 1.5 的隨機浮點數，代表 M2 引入的不可預期性。
4. 執行修正行動：將結果寫入 \text{M}_6 治理核心的日誌中。

# III. 程式與介面的連接 (Code-to-UI Flow)

## A. 技術選型與持久化

- 技術棧：React/JSX with Tailwind CSS (單一檔案架構)。
- 狀態管理：React Hooks (useState, useContext)。
- 資料庫：Google Firestore (使用 __firebase_config 和 __initial_auth_token)。

## B. 核心資料流程 (React - Firestore)

1. 初始化與驗證：
    - 在 App 元件的 useEffect([]) 中初始化 Firebase 和 Auth。
    - 使用 signInWithCustomToken 或 signInAnonymously 完成登入。
2. 實時資料同步：
    - 在登入成功後，使用 useEffect([userId, db]) 啟動所有核心文件的 onSnapshot 監聽器。
    - 核心監聽文件：StarDeityCore、ResourceState、EntityList。
    - 當 Firestore 資料更新時，立即呼叫 setState 更新 React 狀態，觸發 UI 重新渲染。
3. 寫入操作：
    - 所有玩家操作（建造、轉化、升級 M-值）都透過非同步函式 (e.g., updateMValues(newValue)) 寫入 Firestore。
    - 避免重複寫入：在寫入前應進行本地樂觀更新（Optimistic Update）並加入錯誤處理。

## C. 可修改元素 (設置與升級)

以下元素是玩家可主動修改並影響 M-值計算的：

| 可修改元素 | 介面入口 | 影響 M-值 | 邏輯實現 (Firestore) |
|---|---|---|---|
| **M-**值升級 | 星神核心儀表板 | 永久增加 M1-M6 的 base 基礎值。 | 消耗 \text{M}_{\text{Ex}} 異 |

| 可修改元素 | 介面入口 | 影響 M-值 | 邏輯實現 (Firestore) |
| --- | --- | --- | --- |
| | | | 構資源，更新 \text{M}_n \text{ Base} 欄位。 |
| 造物放置/移除 | 建造表單 / 六角網格 | 增加/減少區域 M-值 modifier 修正值。 | 更新地塊文件 (HexTile Document)，並觸發全局 M-值重新計算。 |
| 實體派遣/召回 | 實體清單 / 六角網格 | 增加/減少實體所在區域的 M-值 modifier 修正值。 | 更新實體文件 (Entity Document) 的 location 欄位。 |
| 資源轉化率 | 資源系統下拉菜單 | 調整 \text{M} \to \text{L} 的輸入量。 | 調整 ResourceConversionSettings 文件中的 \text{Input M} 參數，影響下一遊戲循環。 |

# IV. 技術與性能優化 (Performance Optimization)

1. 六角形渲染：優先使用 SVG 渲染六角形，以便於互動和自訂樣式。對於大規模網格，考慮使用 \text{Canvas} 進行效能優化。
2. 狀態顆粒度：避免將所有狀態放在一個巨大的 useState 中。應將核心 M-值、資源、地塊等狀態分拆，以減少不必要的 UI 重新渲染。
3. 移動端優化：確保所有按鈕/互動目標的觸控尺寸 (48 \times 48 \text{px}) 滿足行動裝置標準。

這份規格書涵蓋了遊戲實作所需的架構、數據和計算細節。我們隨時可以開始實作第一個 **React** 原型，從 UI 的佈局和核心 M-值儀表板開始！