Project: CARGO

—— 基于大语言模型的火星生存远程协作模拟终端

Product Whitepaper v1.0

---

1. 产品概述 (Executive Summary)

Project CARGO 是一款文本驱动的生存模拟游戏。

玩家扮演地球指挥中心（Mission Control）的首席调度员，通过唯一的低带宽文字终端，指导滞留在火星废弃飞船上的幸存者 **Jack** 进行自救。

核心创新点在于“非对称能力”：玩家拥有全知视角和技术文档，但无行动能力；NPC（由 AI 驱动）拥有行动能力，但缺乏专业知识且情绪不稳定。游戏的本质不是解谜，而是用自然语言解决**“**认知偏差**”**与**“**情绪摩擦**”**。

---

2. 世界观与角色设定 (Narrative & Persona)

2.1 背景故事

- 时间/地点：2088年，火星，Cydonia 废弃基地。
- 事件：大撤离中发生事故，唯一的留守者并非科学家，而是一名负责搬运货物的后勤搬运工。
- 现状：他躲进了一艘冷战时期的初代飞船（全模拟电路，无自动化），通讯模块损坏，只能收发 ASCII 文本。

2.2 核心角色：**Jack (The Survivor)**

由经过 Prompt Engineering 深度调优的大模型扮演。

- 职业属性：蓝领工人，不懂物理，不懂电路，只认识颜色、形状和简单的机械结构。
- 性格特征：话痨（Copious Talker），压力大时会语无伦次，无聊时会讲烂笑话，对玩家有强烈的情感依赖。
- 行为逻辑：
  - 服从性：高（但他听不懂术语）。
  - 执行力：低（需要极其具体的步骤）。
  - 自我保护：会质疑高风险指令（“你确定要我剪红线？电影里剪红线都炸了！”）。

---

3. 核心游戏循环 (Core Gameplay Loop)

1. 危机触发 **(Trigger):** 后端仿真引擎抛出异常（例：室内温度降至 -10°C）。
2. 信息获取 **(Analysis):** 玩家查阅左屏的《飞船技术维护手册》（PDF/图文），咨询右屏的专家 Agent。
3. 指令构建 **(Input):** 玩家将复杂的技术步骤（“重置加热器继电器”）转化为自然语言（“把左边那个带火焰标志的开关推上去”）。
4. **AI** 理解与反馈 **(Processing):**
    - Jack 尝试理解指令。
    - *If* 指令清晰: Jack 执行 -> 调用后端函数 -> 状态更新。
    - *If* 指令模糊: Jack 困惑 -> 发起反问（“哪一个？这里有两个红开关！”）。
    - *If* 情绪崩溃: Jack 拒绝执行 -> 玩家需先进行安抚（Talk Down）。
5. 仿真结算 **(Simulation):** 后端物理引擎计算结果（成功/故障/受伤），并生成新的环境描述反馈给玩家。

- 盲盒执行 **(Blind Execution)**
    - 玩家发出指令后，系统界面不会有任何变化（无进度条、无弹窗）。
    - 后台静默运行物理计算。
- 感官反馈 **(Sensory Feedback)**
    - 唯一反馈源：Jack 的聊天气泡。
    - 翻译机制：后端将 `State_Change` (状态变更) 转化为一段 `Sensory_Prompt` (感官提示词) 喂给 Jack。
        - 后台事实: `Door_Status = Open`
        - 给*Jack*的提示: "你面前的厚重金属门发出液压泄气的声音，然后缓缓滑开了。"
        - *Jack*生成的回复: "这就开了？这也太容易了，我还以为要像电影里那样用炸药呢。"
    - 信息失真：基于 Jack 的当前状态（恐惧/缺氧/兴奋），他对事实的描述可能会夸大、遗漏或错误。玩家需要自行判断真伪。

---

4. 功能模块详述 (Module Breakdown)

模块 **A:** 玩家前端 **- "三屏指挥台" (The Interface)**

UI 设计模拟真实的指挥中心终端。

1. 中央主控区 **(Comm Link):**
    - 形式：滚动式聊天窗口，复古绿色字符风格。
    - 功能：输入指令，接收 Jack 的回复，显示系统自动生成的“连接状态/延迟”提示。
2. 左侧资料区 **(Technical Library):**
    - 形式：可交互的 PDF 阅读器 / Wiki。
    - 内容：包含《电路图鉴》、《化学合成表》、《应急医疗指南》。
    - 关键点：信息必须是视觉化的（图示为主），强迫玩家进行“看图说话”。
3. 右侧智囊团 **(Advisor Agents):**

- **Agent A (Dr. Logic):** 基于 RAG 技术，能够瞬间检索手册内容，提供最优解，但语气冷漠，不顾 Jack 死活。
- **Agent B (Dr. Empathy):** 心理学专家，分析 Jack 的情绪状态，提示玩家何时该安抚，何时该严厉。

模块 B: AI 智能体架构 (The Brain)

这是项目的技术核心，包含两个层面：

1. **Jack Persona Engine (人格引擎):**
   - **System Prompt:** 注入“无知”、“恐惧”、“依赖”等特质。
   - **Context Filter:** 过滤掉所有专业术语。如果玩家输入“开启阀门”，Jack 会回复“什么是阀门？是这个转盘吗？”。
   - **Emotional State Machine:** 维护一个 Stress_Level 变量。
     - Low Stress: 理解力 90%，说话风趣。
     - High Stress: 理解力 30%，容易产生幻觉（Hallucination Injection），只接受短句指令。
2. **Intent Interpreter (意图识别器):**
   - 一个中间层 LLM，负责将玩家的自然语言翻译为后端代码。
   - *Player:* "把蓝色的推上去。" + *Context (当前场景是配电箱)*
   - *Interpreter Output:* { "action": "toggle_switch", "target_id": "switch_blue_01", "value": "on" }

模块 C: 后端物理仿真 (The Physics)

不追求极致真实的物理，追求“逻辑闭环”的游戏性。

- 状态管理 **(State Management):** 使用 JSON 维护飞船状态。

```JSON
{
  "oxygen": 85,
  "temp": 18,
  "power_grid": {
    "main_bus": false,
    "backup_battery": true
  },
  "jack_location": "cockpit"
}
```

- 规则引擎 **(Rule Engine):**
  - if temp < 0 and heater == off: 每分钟 Jack 生命值 -5。
  - if oxygen < 20: Jack 进入“缺氧状态”，自动在回复中插入乱码或错别字。
  -

C 后端物理仿真 (The Physics) - Sensory Layer Update

在此模块中增加 **“感官转译层” (Sensory Translator)**：

- 功能：将冷冰冰的代码变量转化为人类的感官体验（视觉、听觉、嗅觉、触觉）。
- 映射表 **(Mapping Example):**
    - `Temp > 40°C` -> Prompt: "你感觉汗水流进眼睛里了，呼吸的空气是烫的。" -> Jack: "热死我了，这里像个烤箱！"
    - `O2 < 10%` -> Prompt: "你的视野边缘开始变黑，手脚发麻。" -> Jack: "我觉得...有点困...刚才你说什么来着？"
    - `Short_Circuit (短路)` -> Prompt: "你闻到了焦糊味，听到了啪啪的爆裂声。" -> Jack: "卧槽！什么东西烧焦了？我是不是搞砸了？"

---

5. 游戏流程与关卡规划 (Level Design)

- **Tutorial: The Awakening (觉醒)**
    - 目标：建立联系，确认身份。
    - 操作：引导 Jack 打开手电筒（教会玩家如何描述物体位置）。
- **Level 1: Dark & Cold (饥寒交迫)**
    - 目标：重启电力系统。
    - 难点：配电箱标签模糊，玩家需通过描述线缆颜色和走向来指导接线。
- **Level 2: The Signal (信号)**
    - 目标：校准天线角度。
    - 难点：涉及简单的几何计算，Jack 分不清顺时针逆时针，玩家需用“像表针转动的方向”来解释。
- **Level 3: The Chemistry Set (绝命毒师)**
    - 目标：合成燃料。
    - 难点：高风险操作，一旦指令错误直接 Game Over。需要极高的信任度和情绪安抚。

---

6. 技术栈建议 (Tech Stack)

- **Frontend:** React / Vue (构建三屏布局)。
- **LLM Integration:** OpenAI API / DeepSeek API (支持 Function Calling)。
- **Backend:** Python (FastAPI/Flask) 处理游戏逻辑与状态机。
- **Database:** 简单的 Redis 或 内存存储 (保存当前会话状态)。
- **RAG:** LangChain + VectorDB (用于右侧专家助手检索 PDF 文档)。

💭 深度激荡（继续）

这份白皮书把我们的想法从“点子”变成了“工程图纸”。

我特别想强调一下 **4.B** 模块中的 **Intent Interpreter (意图识别器)**。这是你这个项目能否成功的

关键。

一般的 AI 游戏是：玩家说话 -> AI 说话。

你这个游戏是：玩家说话 -> AI (Interpreter) 分析 -> 修改物理世界 -> AI (Jack) 看到变化后说话。

这个闭环（Human-in-the-loop Simulation）才是真正硬核且有技术含量的部分。

为了让你更直观地理解这个架构，我给你生成一张系统架构图。

📌 落地归档卡

维度	核心内容
文档产出	**Project HANDHOLD Whitepaper v1.0** (已生成)
核心定义	一款 "自然语言驱动的物理仿真游戏"。玩家用语言作为编译器，去编译 Jack 这个充满 Bug 的执行程序。
关键差异	与其说是在玩游戏，不如说是在进行 **Prompt Engineering** 实战演练。玩家必须学会如何向一个不懂技术的大模型清晰地描述需求。
下一步 **(Next Step)**	白皮书有了，架构图也有了。我们现在需要验证最核心的互动体验。 我建议你先别写后端，先用 Prompt 在 ChatGPT 里模拟一下 **"Level 1:** 配电箱" 的场景。 你想让我为你生成这个测试用的 **Prompt** 吗？