Stancelore - MVP 实现方案

技术决策

放弃 Ikemen GO，全部使用 Godot 4。

理由：

- 游戏有大量格斗之外的系统（探索、养成、UI），Ikemen GO 只擅长对战
- 角色全部原创，MUGEN 社区资源复用优势不存在
- Godot 4 架构统一，避免双引擎胶水代码
- Godot 原生支持 Steam 导出，社区活跃
- GDScript 上手快，适合快速原型

技术栈

层面	选型
引擎	Godot 4.x
语言	GDScript（主）+ 可选 C# 热路径优化
2D 格斗	Godot AnimationPlayer + Area2D 碰撞体
AI 系统	自研决策概率表（纯 GDScript）
数据格式	JSON（招式/属性/关卡/NPC 配置）
像素美术	Aseprite（AI 辅助生成 + 手工调整）
版本控制	Git + GitHub
发行	Steam（Godot Steamworks 插件）

项目结构

```
stancelore/
├── project.godot
├── assets/
│   ├── sprites/          # 角色 sprite sheets
│   │   ├── hero/         # 主角各动画帧
│   │   └── enemies/      # 各区域敌人
│   ├── ui/               # UI 素材
│   ├── maps/             # 节点地图背景
│   ├── portraits/        # 角色头像/立绘
│   └── audio/            # 音效和 BGM
├── data/
│   ├── moves/            # 招式定义 JSON
│   │   ├── templates.json    # 基础模板
│   │   ├── affixes.json      # 词缀定义
│   │   └── rare_moves.json   # 稀有招式
│   ├── enemies/          # 敌人配置 JSON
│   ├── stages/           # 关卡/区域配置
│   └── dialogues/        # NPC 对话文本
├── scenes/
│   ├── battle/           # 战斗场景
│   │   ├── battle_scene.tscn
│   │   ├── fighter.tscn      # 格斗角色基础场景
│   │   └── hud/              # 战斗 UI
│   ├── exploration/      # 探索场景
│   │   ├── world_map.tscn    # 世界地图
│   │   ├── node_map.tscn     # 区域节点地图
│   │   └── town.tscn         # 据点/城镇菜单
│   ├── management/       # 养成管理
│   │   ├── character_sheet.tscn  # 属性/加点
│   │   ├── move_inventory.tscn   # 招式管理
│   │   └── move_equip.tscn      # 招式装备
│   └── main/
│       ├── title.tscn
│       └── main.tscn
├── scripts/
│   ├── battle/
│   │   ├── fighter_controller.gd   # 角色战斗控制
│   │   ├── ai_brain.gd             # AI 决策核心
│   │   ├── decision_table.gd       # 决策概率表
│   │   ├── move_executor.gd        # 招式执行
│   │   ├── hitbox_manager.gd       # 碰撞判定
│   │   └── coach_system.gd         # 教练指令
│   ├── data/
│   │   ├── move_data.gd            # 招式数据结构
│   │   ├── move_generator.gd       # 程序化招式生成
│   │   ├── character_stats.gd      # 角色属性
```

```
│   │   └── save_manager.gd          # 存档
│   ├── exploration/
│   │   ├── map_manager.gd           # 地图逻辑
│   │   ├── encounter_system.gd      # 遭遇战
│   │   └── npc_dialogue.gd          # NPC 对话
│   └── core/
│       ├── game_manager.gd          # 全局状态
│       └── event_bus.gd             # 事件总线
└── addons/                  # 第三方插件
```

开发阶段划分

Phase 0：原型验证（2-3 周）

> 目标：验证”AI 概率决策对战 + 教练指令”是否好玩

只做战斗核心，其他全部跳过。

- 两个方块（placeholder）在屏幕上对打
- 实现决策概率表，能根据距离/血量选择行为
- 行为先做 4 种：攻击、防御、回避、待机
- 每种行为有简单的动画（方块变色/位移即可）
- 命中判定用 Area2D 碰撞
- 打击反馈：命中暂停帧（2-3 帧冻结）+ 屏幕震动
- 实现教练指令（3 种就够：猛攻/均衡/防守）+ 5 秒冷却
- 简单的 HP 条 UI
- 硬编码 6 个属性值，手动改数值测试不同加点的 AI 行为差异

验收标准： 两个方块打起来有节奏感、有来回，切换教练指令能明显感到行为变化。如果这个都不好玩，后面的一切都没意义。

Phase 1：战斗系统完善（3-4 周）

> 目标：完整的格斗对战体验

1.1 角色动画替换

- 用 Aseprite 制作主角的基础 sprite sheet
- 最小动画集：idle / walk / 3种攻击 / 受击 / 倒地 / 防御 / 回避
- 每个动画 4-6 帧，像素风格（32x32 或 64x64）
- 可以先用 AI 工具（如 PixelLab）生成初稿再手工调整

1.2 招式系统实现

```
# move_data.gd 核心数据结构

class_name MoveData extends Resource

enum Type { ATTACK, DEFEND, EVADE, GRAB, SPECIAL }
enum Range { CLOSE, MID, FAR }

@export var id: String
@export var template_name: String        # 基础模板名
@export var type: Type
@export var range: Range
@export var damage: float
@export var speed: float                 # 出招速度（帧数）
@export var recovery: float              # 收招硬直（帧数）
@export var hit_stun: float              # 给对手的硬直
@export var knockback: Vector2           # 击退方向和力度
@export var affixes: Array[AffixData]    # 词缀列表
@export var rarity: int                  # 0白 1蓝 2紫 3金
@export var animation_key: String        # 对应的动画名
```

1.3 程序化招式生成器

- 读取 templates.json 获取基础模板
- 读取 affixes.json 获取词缀池
- 按品质等级随机组合：白色1词缀、蓝色2词缀、紫色3词缀
- 词缀修改招式的数值属性（伤害倍率、附加效果等）

1.4 AI 决策系统完善

```
# ai_brain.gd 核心逻辑（伪代码）

func decide_action(state: BattleState) -> Action:
    # 1. 获取当前状态对应的基础概率表
    var table = decision_table.get_probabilities(
```

```
        state.distance,
        state.self_hp_ratio,
        state.opponent_state
    )

    # 2. 属性修正
    table = apply_stat_modifiers(table, character_stats)

    # 3. 教练指令修正
    table = apply_coach_directive(table, current_directive)

    # 4. 归一化
    table = normalize(table)

    # 5. 按概率随机选择行为类别
    var action_type = weighted_random(table)

    # 6. 从该类别已装备的招式中选一个
    var move = pick_move(action_type, equipped_moves)

    return move
```

1.5 打击感强化

- 命中暂停帧（hit stop）：3-5 帧
- 屏幕震动：根据伤害大小调整幅度
- 击中火花特效（简单粒子）
- 受击动画 + 击退位移
- 音效（拳击声、防御声、受击声）

Phase 2：养成系统（2-3 周）

> 目标：属性加点 + 招式管理完整可用

2.1 属性系统 UI

- 六维属性显示（STR/AGI/VIT/INT/DEX/LUK）
- 加点/减点按钮，实时预览 AI 行为倾向变化
- “全部重置”一键清空重分配
- 当前等级和可用点数显示

2.2 招式管理 UI

- 招式背包列表（显示名称、类型、品质颜色、词缀标签）
- 8 个装备槽位
- 拖拽或点击装备/卸下
- 招式详情面板（数值、词缀效果说明）
- 按类型/品质/元素筛选和排序

2.3 升级系统

- 对战胜利/失败获得经验值（胜利更多）
- 升级动画和提示
- 每级获得 5 个属性点

2.4 招式领悟系统

- 被动领悟：被命中时概率触发，对战结束后结算显示
- 压力突破：血量 < 10% 时概率触发
- 对战结束结算画面：经验获得 + 招式领悟展示

Phase 3：探索系统（2-3 周）

> 目标：节点地图 + NPC + 3 个区域可游玩

3.1 世界地图

- 简单的 2D 地图画面，标注各区域位置
- 已解锁区域可点击进入
- 未解锁区域显示为迷雾/锁定状态 + 解锁条件提示

3.2 区域节点地图

```
# stages/forest.json 示例

{
  "id": "magic_forest",
  "name": "魔法森林",
  "description": "浓雾弥漫的古老森林，传说中自然魔法的发源地...",
```

```
"nodes": [
  {
    "id": "forest_entrance",
    "type": "battle",
    "name": "森林入口",
    "description": "几只魔物在入口处徘徊。",
    "enemy_id": "forest_slime",
    "connections": ["hunters_lodge", "poison_trail"],
    "unlock_condition": null
  },
  {
    "id": "hunters_lodge",
    "type": "town",
    "name": "猎人小屋",
    "description": "一个独居的老猎人在此歇脚。他似乎知道很多森林深处的秘密。",
    "npcs": ["old_hunter"],
    "connections": ["deep_forest"],
    "unlock_condition": null
  },
  {
    "id": "poison_trail",
    "type": "battle",
    "name": "毒沼小径",
    "description": "空气中弥漫着刺鼻的气味，脚下的泥土呈现不自然的紫色。",
    "enemy_id": "poison_frog",
    "connections": ["hidden_shrine"],
    "unlock_condition": null
  },
  {
    "id": "hidden_shrine",
    "type": "discovery",
    "name": "???",
    "description": "藤蔓后面似乎有什么东西...",
    "reward": "scroll_fragment_nature_01",
    "connections": [],
    "unlock_condition": {"type": "move_element", "element": "fire"}
  },
  {
    "id": "deep_forest",
    "type": "battle",
    "name": "森林深处",
    "description": "巨大的古树遮蔽了天空，一个身影在树间修炼。",
    "enemy_id": "forest_monk",
    "connections": ["boss_guardian"],
    "unlock_condition": null
  },
  {
```

```
      "id": "boss_guardian",
      "type": "boss",
      "name": "森之守护者",
      "description": "守护这片森林千年的精灵战士。他的自然拳法据说能让大地震颤。",
      "enemy_id": "forest_guardian",
      "connections": [],
      "unlock_condition": {"type": "level", "min_level": 10}
    }
  ]
}
```

- 节点地图用简单的连线图渲染
- 当前位置高亮，可移动到相邻已解锁节点
- 每个节点点击显示文字描述
- 战斗节点：进入后触发对战
- 据点节点：弹出 NPC 对话菜单
- 隐藏节点：满足条件后才显示

3.3 NPC 对话系统

- 简单的文字对话框 + 选项
- 对话内容从 JSON 读取
- NPC 提供情报（稀有招式线索、隐藏路径提示）
- MVP 阶段不需要立绘，用简单的名字标签即可

3.4 MVP 包含的 3 个区域

1. **始まりの村**：教学区域，引导玩家熟悉系统
2. **魔法森林**：自然/毒系招式，5-6 个节点 + 1 个 Boss
3. **王国废墟**：剑术系招式，5-6 个节点 + 1 个 Boss

Phase 4：整合打磨（2-3 周）

> 目标：完整可玩的游戏循环 + Steam 上架准备

4.1 流程串联

- 标题画面 → 新游戏/继续 → 世界地图 → 区域探索 → 对战 → 结算 → 循环

- 存档/读档功能
- 新手引导（前几场战斗的教学提示）

4.2 敌人制作

- 每个区域 3-4 种普通敌人 + 1 个 Boss
- 每个敌人需要：sprite sheet、属性配置、招式列表、AI 概率表
- Boss 需要精心设计属性+招式搭配，体现流派特色
- MVP 总计约 8-10 个敌人角色

4.3 数值平衡

- 属性成长曲线
- 招式伤害/速度平衡
- 敌人难度递进
- 经验需求曲线
- 招式掉落率

4.4 UI 打磨

- 统一的像素风 UI 主题
- 战斗中教练指令按钮 + 冷却显示
- 招式领悟的演出效果（发光、音效）
- 升级演出

4.5 Steam 上架

- Steamworks SDK 集成（成就、云存档）
- 商店页面素材（截图、视频、描述）
- Steam Deck 兼容测试
- 构建 Windows / macOS / Linux 版本

美术资产清单（MVP 最小集）

资产	数量	规格
主角 sprite sheet	1	64x64，约 15 个动画 × 4-6 帧
敌人 sprite sheet	8-10	64x64，约 10 个动画 × 4-6 帧
战斗背景	3-4	像素风场景画面
招式特效	10-15	粒子/帧动画
节点地图 UI	若干	节点图标、连线、背景
世界地图	1	简化大陆俯视图
UI 组件	若干	按钮、面板、字体、图标
音效	20-30	打击、防御、受击、UI 交互等
BGM	4-5	标题、战斗、探索、Boss、城镇

时间线总览

阶段	时间	核心交付
Phase 0 原型验证	第 1-3 周	方块对打原型，验证核心玩法
Phase 1 战斗完善	第 4-7 周	完整格斗系统 + 像素角色 + 打击感
Phase 2 养成系统	第 8-10 周	属性加点 + 招式管理 + 升级
Phase 3 探索系统	第 11-13 周	节点地图 + NPC + 3 个区域
Phase 4 整合打磨	第 14-16 周	流程串联 + 平衡 + Steam 上架

总计约 4 个月（假设每天投入 3-4 小时）

Phase 0 立即开始的步骤

1. 安装 `Godot 4.4`（最新稳定版）
2. 创建项目 `stancelore`
3. 新建 `battle_scene.tscn`
4. 创建两个 `ColorRect` 作为 `placeholder` 角色

5. 实现 `fighter_controller.gd`：
   - `idle / walk / attack / defend / evade` 状态机
   - `Area2D` 碰撞检测
6. 实现 `decision_table.gd`：
   - 距离判定（近/中/远）
   - 基础概率表（攻击40%/防御25%/回避15%/反击10%/待机10%）
7. 实现 `ai_brain.gd`：
   - 读取概率表 → 加权随机 → 执行行为
8. 实现 `coach_system.gd`：
   - 3 个按钮（猛攻/均衡/防守）
   - 5 秒冷却 `Timer`
   - 点击后修改概率表偏移
9. 添加 `hit stop`（命中暂停帧）和屏幕震动
10. 测试：调整概率数值直到两个方块打起来好看

风险和注意事项

风险	应对
AI 对战不好看	Phase 0 最优先验证，不好看就调概率和行为节奏
像素美术工作量大	优先用 AI 工具辅助，先求统一风格再求精美
格斗手感差	重点投入 hit stop、屏幕震动、音效，这三个决定手感
数值平衡困难	先粗调跑通，上 Early Access 后根据玩家反馈迭代
一个人开发容易倦怠	Phase 0 快速出原型获得正反馈，每阶段有明确交付物