

一、地图生成子系统设计

1. 设计目标与总体思路

1.1 设计目标：

本子系统将基于 txt 的拼块地图，在运行时按需转换为可见与可交互的关卡内容，并通过“按段流式生成”的方式支撑持续下落的长关卡体验。

核心目标包括：在生成期避免一帧内大量实例化导致的卡顿；保证竖直与水平两个阶段的无缝衔接；通过“进入新段顶部即删除旧段”的生命周期管理控制场景体量与内存；通过层级隔离与暂停策略，确保 UI 与演出不受地图生成与清理影响。

1.2 总体思路与模块分工：

系统拆分为三层协同：MapDecider 负责读取与解析地图文本、按格异步发射为“字符+坐标”；MapSpawner 负责将字符映射为具体 PackedScene、实例化并挂段容器的 Tiles 和 Enemies；Master 负责段级调度，预建触发与删旧段。全流程采用异步分帧接口，保证帧率稳定。

2. 数据源与字符映射

2.1 地图文本格式

地图数据位于 Data 中各个 map.txt 文件中，每个文件由“=”分隔为多个 chunk，每个 chunk 固定为 5×17 。MapDecider 读取后缓存为字典，并在遍历时将列号统一转换为 (col-1)，使模板最左的墙列成为逻辑列-1，配合 `off_set_x=16` 令左墙像素对齐世界 $x=0$ ，确保竖向主干的左界线稳定。

2.2 难度数据集与切换

系统维护五套数据集。Master 按段号实现难度提升与主题切换。MapDecider 在 `_ready` 中完成数据装载并设置当前数据指向。

2.3 字符到实体映射（不完整，等 cly 加完那些新的怪再改）

MapSpawner 中的 `spawn_block()` 将字符映射到 `PackedScene` 并实例化。主要类别包含：地形/装饰、敌人、道具、宝石。为便于段级回收，静态与道具统一落在 `Tiles`，敌人统一落在 `Enemies`。必要的行为在生成时连接，例如“刺”节点直接连接到 `Master.hurt_player`。

2.4 颜色效果同步与特例

节点生成后通过红/黄/蓝/彩/绿五个效果管理器更新着色状态；对不应被染色的对象（如最后的狗儿）通过分组与元数据标记进行排除。字符“G”生成后会被移动至 `Master` 下的独立 `CanvasLayer`（更高绘制层），确保其不会被随段清理且不被全屏覆盖层影响。

3. 坐标体系与对齐策略

3.1 列偏移与左墙对齐

为消除模板至世界坐标的系统性偏移，遍历采用“逻辑列 (`col-1`)”规则，将文本最左一列视为-1，并配合 `off_set_x=16`，使左墙像素恰对齐世界 `x=0`，从而保证竖向主干在 `x` 方向上的一致性。

3.2 偏移与锚点

竖向阶段的 `x` 锚点由 `vertical_origin_x` 提供，发射坐标为 `(vertical_origin_x + colTILE, off_set_y + rowTILE)`；横向阶段使用 `off_set_x` 作为列起点并按可见列数推进。

每生成一个竖向块 `off_set_y` 增加 `CHUNK_H`；横向尾部拼出若干列后，记录最后一列的起始 `x`，并将 `vertical_origin_x` 锁定至该值。随后 `off_set_y` 额外下推 `2*CHUNK_H`，让下一段竖向主干从三层底部启动，避免与本段的尾部相互干扰。

4. 段的流式生成

4.1 段的定义与容器管理

每一段由“竖向主干+横向尾部+锁锚点+下推起点”构成。`Master` 在建段时创建 `Segment_X` 根节点，内含 `Tiles` 与 `Enemies` 两个子节点；调用

`MapSpawner.set_spawn_roots(tiles, enemies)` 暂时覆盖生成父节点，使该段内所有对象都被收拢到段容器下。段生成完成后恢复默认父节点，以免影响其他系统的实例化路径

4.2 段内生成序列（异步）

`spawn_one_segment_async` 按固定序列异步分帧发射：第一段先生成起始块，随后堆叠若干空白块、随机块与特定编号块，期间每生成一个块，`off_set_y` 就递增 `CHUNK_H`。竖向阶段仅使用异步接口以避免单帧发射过量造成卡顿。

4.3 横向尾部与锚点锁定

横向阶段在同一高度上拼接若干横向列，采用 `make_specific_chunk_right_stack3_async(...)` 等接口一系列三层地铺设，并多次调用形成水平通道或平台。并记录最后一列起始 `x`。水平阶段结束后，调用锁定接口将 `vertical_origin_x` 设为最后一列的起始 `x`，确保本段横向到下一段竖向的衔接顺利。

4.4 段边界统计

段生成完成后，`Master` 会对段容器子树中的 `Node2D` 进行一次粗略遍历，统计 `global_position.y` 的最小值与最大值，作为该段的 `ymin` 与 `ymax`。用于预建触发（接近 `ymax`）、删旧触发（进入新段 `ymin`），成本低且精度满足需求。

5. 异步分帧与性能

5.1 分帧预算与逐格发射

`MapDecider` 以 `SPAWN_BUDGET`（默认 400 cell/帧）为限制，在达到预算后 `await` 一帧，从而将实例化压力拆散到多帧，竖向主干与横向尾部均以异步分帧完成，玩家几乎不会感知到卡顿。

5.2 预生成与删除时机

Master 会在玩家的 y 超过当前段 $y_{\max} - \text{BUILD_TRIGGER_MARGIN}$ （默认为 $2 * \text{CHUNK_H}$ ）且不在建段、未达段上限时，立即开始异步构建下一段；而当玩家的 $y \geq$ 新段顶部的 y_{\min} 时，立即删除上一段容器，保证场上几乎只有当前段，减少性能开销

6. 可扩展性

新增关卡只需要在 Data 中添加或调整模板（只要严格保持 5×17 ）；新增元素只需在 MapSpawner 的字符映射中登记并预载对应 PackedScene，即自动被发射路径覆盖。段的内部生成节奏可通过调整 `spawn_one_segment_async` 中的序列与数量实现。