一个不留---开发 Tips

适用场景：HTML5 单文件试玩广告 / 横竖屏自适应小游戏 · 投放 Google Play（AdMob / Google Ads）
参考项目：三阶段交互 + 拆螺丝物理小游戏（Last Light）

一、提示词（需求文档）的完整性与详细性（提示词在最后）

从「让 AI / 协作者一次写对」的角度，有几类关键信息必须在提示词中前置明确，否则极易导致大规模返工：

1.1 必须明确的「全局约定」（最容易遗漏）

维度	说明	项目当前情况
1	坐标系基准：是用百分比设计坐标还是绝对像素？孔位匹配规则是什么？	✅ 后期补充了「设计坐标系 + 百分比」，初版未明确导致孔位逻辑返工
2	实体独立性：各核心元素的从属/独立关系	⚠️ 本次最大返工点："螺丝插回孔，铁板跟着过来"就是因为初版未讲清「铁板与螺丝是独立个体」
3	跳转方式：是 location 跳转还是 Exit API？	✅ 后期明确，但 APL 渠道必须开篇声明
4	资源形态：外链 / 同级 asset /	✅ 已分阶段说明

	base64 内联	
5	语言版本：简体 / 繁体 / 多语言	✅ 繁体台湾版已单独说明

核心Tips：写提示词时，先用一句话明确「实体关系」，讲清谁依赖谁、谁独立，能避免至少一轮大返工。例如：

"背景铁板（含孔）是底图固定层；铁板是独立刚体；螺丝是独立刚体。螺丝只与背景孔交互，铁板孔仅用于开局定位与脱落检测，三者不绑定。"

1.2 提示词结构建议（可复用模板）

- **【投放渠道与合规】**明确渠道、单文件要求、跳转方式规则
- **【屏幕适配】**优先级、适配规则、安全区要求
- **【素材清单】**列全所有素材格式 + 横竖屏变体命名规则
- **【坐标系约定】**坐标基准、孔位计算规则
- **【实体关系】**(关键!) 明确谁独立、谁依赖、交互边界
- **【玩法状态机】**阶段流转规则 + 胜负条件 + 死局兜底逻辑
- **【代码规范】**模块要求、命名规则、性能要求
- **【埋点/CTA】**触发跳转的区域、链接占位规则

二、框架搭建（架构层面的关键决策）

2.1 单 Canvas + DOM 分层混合架构

本次项目采用的分层架构清晰合理，值得复用：

#game 固定全屏容器

├── bgImg / bgVideo 交叉淡入（背景层）

├── overlay 选中图覆盖（UI层）

├── choiceLayer DOM 选项（UI层）

├── screwLayer + Canvas 拆螺丝玩法（核心玩法层）

├── settleLayer DOM 结算（UI层）

├── brandLogo 品牌角标（UI层）

└── capsuleSafe 右上胶囊安全区（预留层）

- 背景 / UI / 结算用 DOM：自适应、动画可直接用CSS实现，开发效率高，维护省心
- 物理玩法用 Canvas：高频重绘、碰撞检测在Canvas中性能更稳定

这种"DOM管布局，Canvas管玩法"的分工是试玩广告的最佳实践，不建议把所有内容全塞Canvas，也不建议全用DOM实现。

2.2 状态机驱动模式

本次项目用state变量贯穿全局，状态流转为：idle → choice → screw → video → settle，这个模式非常推荐复用。

核心Tips：每个交互入口第一行都添加判断if (state !== "xxx") return;，是防止双击穿透、动画期间误触的关键，需要全程保持这个规范。

2.3 物理实现方案选择：自研简化物理 vs Matter.js

本次项目最初计划使用Matter.js实现物理效果，最终采用自研轻量物理方案（重力 + 悬挂转动 + 倾倒 + 多边形碰撞），该方案实际更适合试玩广告场景，对比如下：

对比维度	Matter.js	自研轻量方案
体积	增加 ~85KB（min版）	0额外体积（纯手写逻辑）
APL体积合规性	体积压力大	✅ 对体积限制友好
异形板碰撞支持	需要手动创建凹多边形碰撞体	✅ 可通过自定义多边形参数直接实现
可控性	物理求解器偶尔会出现不符合预期的结果	✅ 完全可控，可精确实现死局判定等逻辑

结论：试玩广告优先选择自研简化物理方案，不建议引入Matter.js等重型物理引擎。本次使用的hanging / tipping / falling 三态 + 多边形SAT碰撞方案已经满足需求，可继续维护复用。

三、代码运行 & 健壮性优化

3.1 已验证的优秀实践

- ✅ 触摸/点击双绑定 + touched去重，有效防止移动端click幽灵触发
- ✅ requestAnimationFrame驱动渲染，cancelAnimationFrame及时清理
- ✅ 图片decode()完成后再切换背景，避免页面闪白
- ✅ 视频使用loadeddata / playing / play().then 多重保险保证正常露出

3.2 建议加固的点

- **音频解锁兜底：**iOS环境下bgm.play()必须在用户手势内首次触发。当前已经在onScreenTap中实现startBgm，但视频结束后恢复bgm的步骤如果用户中途切后台，play()可能被浏览器拦截，建议添加catch后，挂载visibilitychange事件重试。
- **audioCtx 创建时机优化：**当前在playThud中懒加载AudioContext没有问题，但首次拆螺丝才创建会导致第一次音效延迟。建议在首个用户手势中调用audioCtx.resume()完成预热。
- **finish() 重入保护：**死局判定和倒计时归零可能在同一帧都触发finish逻辑，建议添加重入保护：

let finished = false;

function finish(win){

if (finished) return; // ★防重入

finished = true;

running = false; /* ... */

}

// start() 中重置 finished = false;

- **resize 节流：**window.resize事件在窗口拖动时会高频触发，建议添加requestAnimationFrame节流，避免computeView计算抖动。

四、Google APL（试玩广告）合规要点（重点）

合规是试玩广告上架过审的核心命门，需要重点关注：

4.1 跳转规则：只能使用Exit API

当前项目的跳转实现正确：goToStore()优先调用ExitApi.exit()，其次用mraid.open兜底，最后用window.open兜底，没有使用location.href直接跳转，符合规范。

但存在一处隐患：注释说明包名已修改为hkt，但STORE_URL常量仍然是旧包名com.lighthouse.freeplay.eur，根据最新需求，导流目标应为com.lastlight.freeplay.hkt，虽然广告容器内实际以后台配置为准，但兜底跳转仍然会跳错，建议修改为：

const STORE_URL = "https://play.google.com/store/apps/details?id=com.lastlight.freeplay.hkt";

4.2 exitapi.js 必须保持外链

<script src="https://tpc.googlesyndication.com/.../exitapi.js">需要保持外链，不能内联，当前实现正确，继续保持即可。

4.3 声音启动规则

APL规范要求首帧必须静音，首次用户点击后才允许播放声音。当前项目通过bgmStarted锁+首tap启动bgm的实现符合规范，引导文案使用英文也符合渠道要求。

4.4 常见审核红线自查清单

- [ ] 不得自动跳转商店（只能用户点击CTA/Exit按钮触发跳转）
- [ ] 不得伪造系统UI（需要预留关闭按钮位置、胶囊安全区，当前项目capsuleSafe已经预留，符合要求）

☐ CTA按钮需要清晰明显（当前项目banner+双按钮+角标引导，符合要求）

- [ ] 单文件结构，可离线运行（除exitapi.js外不依赖外部网络）
- [ ] 文件体积通常≤5MB（符合AdMob要求）

五、内容压缩问题（重点）

当前项目已经将素材处理为"72张png + 5个mp3 base64内联到单HTML"，体积控制是试玩广告上线的核心难点，以下是可复用的压缩方案：

5.1 体积预算参考

渠道	单文件体积上限（常见标准）
Google Ads / AdMob	5MB（部分渠道要求2MB）
部分DSP	2~3MB

base64编码会让二进制体积膨胀约33%，因此原始素材最好压缩到3.5MB以内再进行内联。

5.2 压缩策略（按收益排序）

图片压缩：

- 将所有PNG转换为WebP，体积通常可以降低50%~70%，当前部分素材已经使用WebP，建议将剩余PNG全部转换

- 静态图使用pngquant / tinypng做有损量化压缩
- 横竖屏变体：评估是否真的需要两套素材，能用object-fit+定位解决的就不要输出双份素材

视频压缩（体积最大来源）：

- 使用H.264 baseline编码，降低码率，分辨率降到720p甚至540p即可满足需求
- 帧率24~30fps足够，不需要更高帧率
- 过场视频时长尽量控制在1~2s
- 横竖屏变体视频是体积杀手，优先使用一套视频+CSS object-fit: cover自适应，实在无法满足需求再输出双份

音频压缩：

- mp3码率降到96~128kbps，单声道即可满足需求
- bgm制作10~20s短循环，不要嵌入整首音乐

base64内联优化：

- 内联后HTML体积较大，开启服务端gzip/brotli压缩（投放平台一般默认开启），base64文本经过gzip压缩后能进一步节省体积
- 发布版单独对代码进行minify压缩，可缩短变量名减少体积

5.3 加载体验优化

当前项目采用分级预加载策略（preloadCritical加载首屏 → schedulePreloadRest空闲后台加载剩余资源），这套方案体验很好，推荐复用：

- 首屏只预热第一关+logo+play按钮，保证秒开
- 其余资源（含横竖屏变体、视频）使用requestIdleCallback在后台加载

不要一次性预加载全部base64资源，会导致首屏卡顿。

六、屏幕横竖屏自适应（重点）

6.1 三层自适应方案（可复用完整方案）

viewport层：禁缩放 + viewport-fit=cover

CSS层：100dvh + vmin/vw单位 + safe-area适配

JS层：isLandscape判断切换横竖屏图/视频变体

Canvas层：computeView按背景宽高比letterbox居中

- CSS层：vmin/vw自适应 + @media (orientation: landscape)重排尺寸 + env(safe-area-inset-*)处理安全区，当前实现到位
- 素材层：提前准备ORI_IMG / ORI_VIDEO横竖屏两套变体，按方向自动获取对应素材

- Canvas层：computeView按背景图宽高比做letterbox居中，toScreen统一处理坐标变换

6.2 值得保留的细节

- 100dvh兜底：当前写了height:100vh; height:100dvh;，旧浏览器回退到vh，新浏览器使用dvh避开移动端地址栏抖动，实现正确
- 方向切换时视频续播：applyOrientationSwap中记录currentTime再换源续播，体验流畅，保留该方案
- 孔位百分比设计：拆螺丝孔位用百分比存储，toX/toY映射设计坐标，横竖屏切换时computeView重算DESIGN_H = DESIGN_W / ar，孔位自动跟随适配，这是正确的实现方案

6.3 潜在问题与优化建议

拆螺丝游戏进行中如果用户旋转屏幕，resize只会重算视图（computeView），buildLevel使用的DESIGN_H会变化，已落位的板子/孔的世界坐标会按新DESIGN_H重算。因为孔位使用百分比存储，孔位适配没有问题，但下落中板子的dx/dy/pivot是绝对设计坐标，旋转瞬间可能出现位置跳动。

建议优化方案：拆螺丝进行中冻结方向切换的重建，或者旋转时对dx/dy也按比例缩放。试玩广告时长短，多数情况下该问题不影响体验，但需要关注这个边界场景。

七、代码规范落地

当前要求「模块开头分隔注释、关键逻辑单行注释、模块化封装」，但当前代码注释偏少，建议统一补充为以下风格：

// —— 资源与常量 ——

const IMG = { /* ... */ };

// —— 屏幕方向工具 ——

function isLandscape() { /* 宽 > 高 即横屏 */ }

// —— 背景层切换（交叉淡入，先盖后撤防闪白）——

function setBgImage(src) { /* ... */ }

// —— 拆螺丝玩法核心 ——

const ScrewGame = (function () {

// —— 碰撞检测：多边形 vs 圆形 ——

function polyCircle() { /* ... */ }

// —— 状态机：fixed → hanging → tipping → falling → gone ——

function afterChange(b) { /* ... */ }

})();

当前使用IIFE模块封装ScrewGame的方式很好，将私有函数封装只暴露start/finish/clearTimer接口，避免内部函数泄露到全局，继续保持该规范即可。

八、发布前自查清单（Checklist）

- ☐ STORE_URL 包名已更新为 com.lastlight.freeplay.hkt
- ☐ exitapi.js 保持外链，其余素材全部base64内联，输出为单文件
- ☐ 单文件体积 ≤ 5MB（gzip前原始素材 ≤ 3.5MB）
- ☐ 首帧静音，首次tap后才出声+启动游戏
- ☐ 所有CTA跳转都走goToStore()，无location.href直接跳转
- ☐ 横竖屏切换测试（覆盖开局/拆螺丝中/过场视频/结算四个场景）
- ☐ 右上160×100胶囊区无内容，顶部底部安全区已预留
- ☐ 禁缩放生效（双指捏合无反应）
- ☐ 第三关死局判定 + 倒计时归零 → 失败结算流程正常
- ☐ 失败retry / play按钮均跳转商店（符合需求要求）
- ☐ 繁体台湾版所有字幕均为繁体（投放台湾市场时检查）
- ☐ 真机测试：覆盖iOS Safari / Android Chrome / 微信内置浏览器
- ☐ finish()已添加重入保护

框架提示词：

基于 HTML5开发一款横竖屏（记得根据尺寸自适应屏幕显示）版简单交互游戏。代码规范要求

注释规范，代码模块化书写

模块开头：// —— 模块名称 ——标注；

关键行为动作逻辑：单行注释说明功能。开发环境

开发工具：VSCode

适配要求：移动端优先（横竖屏、触摸操作）兼容鼠标操作方便电脑调优，适配不同屏幕尺寸，禁用用户缩放，保证触摸无延迟。 性能规范。

下面说的图片，音频和视频的素材都在html文件同级的asset文件夹里，注意引用位置。

游戏设定：点击后屏幕内的待机背景图片为first.png，游戏有背景音乐：bgm.mp3（循环播放）。

1.游戏分为三个阶段：

第一阶段：触发逻辑：玩家点击屏幕即进入第一阶段。场景：屏幕下方会弹出左右两个图片（点击会播放音频：click.mp3。有点击交互逻辑，两张图片大小要一致，一左一右）：feather.png，

finger.png。点击其中任意一个会跳转拆螺丝游戏第一关（后面有该游戏的逻辑设计），当拆螺丝游戏胜利则跳转回来，背景图片换为视频：feather.mp4（视频播放的同时加入刚开始选择的图片feather.png或者finger.png在屏幕中心正下方位置绘制在视频上别遮挡图片，播放一遍后跳转为背景图片：second.png）。

第二阶段：触发逻辑：当背景为图片second.png后再点击屏幕。场景：屏幕下方会弹出左右两个图片（点击会播放音频：click.mp3。有点击交互逻辑，两张图片大小要一致，一左一右）：tentacle.webp，shovel.webp。点击其中一个会跳转拆螺丝游戏第二关（后面有该游戏的逻辑设计），当拆螺丝游戏胜利则跳转回来，如果点击的是tentacle.webp则背景图片换为视频：tentacle.mp4。播放一遍后跳转为背景图片：third.png。如果点击的是shovel.webp则背景图片换为视频：feather.mp4（视频播放的同时加入刚开始选择的图片shovel.webp在屏幕中心正下方位置绘制在视频上别遮挡图片，播放一遍后跳转为背景图片：second.png）。

第三阶段：触发条件：当第二阶段结束后背景为图片second.png或者是third.png后再点击屏幕。场景：屏幕下方会弹出左右两个图片（点击会播放音频：click.mp3。有点击交互逻辑，两张图片大小要一致，一左一右）：durian.png，drill.webp。点击其中一个会跳转拆螺丝游戏第三关（后面有该游戏的逻辑设计），当拆螺丝游戏胜利则跳转回来，背景图片换为视频：feather.mp4（视频播放的同时加入刚开始选择的图片durian.png或者drill.webp在屏幕中心正下方位置绘制在视频上别遮挡图片，播放一遍后跳转为背景图片：second.png）。然后跳转胜利结算界面。

结算界面逻辑（弹出结算界面游戏会暂停）（界面记得做自适应尺寸）：

胜利结算：当玩家完成第三阶段后自动跳转。界面布局：分上中下三张图片：上（无点击交互逻辑）：banner_win.png（这张图片比下面两张都稍微大一些），中（点击交互会跳转链接）：btn_play.png，下（点击交互也会跳转链接）：btn_next.png。

失败结算：当玩家游玩拆螺丝游戏倒计时结束后还未拆完自动跳转界面。界面布局：分上中下三张图片：上（无点击交互逻辑）：banner_fail.png（这张图片比下面两张都稍微大一些，中和下的图片保持大小一致），中（点击交互会重新开始游戏）：btn_retry.png，下（点击交互会跳转链接）：btn_play_fail.png。