Game Design Document (GDD)

太好了，我们可以先用“玩家购买原料 → 合成香水 → 卖给顾客”的流程来设计整体的系统架构。现在我们就进入 **开发步骤规划阶段**，确保你每做完一部分，都能“看到成果”，并及时测试和调整。

**✅ 第 0 步：初始化项目**

**你需要完成：**

* 创建文件夹结构（下一步我们会详细讲）
* 初始化 TypeScript + Phaser 项目（如果你还没做，我可以一步步带你）
* 打开开发服务器，确认能看到一个 Phaser 的游戏画面（哪怕只是黑屏 + 控制台无报错）

**你应该看到的结果：**

* 能打开游戏界面
* 控制台没有报错
* 有基础的游戏循环在运行（一个简单场景）

**✅ 第 1 步：实现游戏主结构（ECS 架构雏形）**

**你需要完成：**

* 创建 ECS 的基本系统框架
  + Entity.ts
  + Component.ts
  + System.ts
* 实现 GameObject 的注册机制

**你应该看到的结果：**

* 控制台打印一些 “Entity Created”、“Component Added” 的日志
* 游戏里可以动态创建一个物品对象（哪怕看不到，也要 log 出来）

**✅ 第 2 步：资源系统（原料）展示**

**你需要完成：**

* 创建一个 **原料列表界面**（比如几个 icon 和名字）
* 玩家可以点击“购买”按钮来购买这些原料
* 原料库存保存在某个系统里，比如 InventorySystem

**你应该看到的结果：**

* 游戏画面显示“香精油”、“酒精”等按钮
* 点击后，控制台打印“购买成功”，并更新库存
* 玩家库存显示在某个角落或 debug log 里

**✅ 第 3 步：合成系统（香水配方）**

**你需要完成：**

* 定义几个简单的香水配方（比如：香精油A + 酒精 = 香水1）
* 玩家选择材料后，可以点击“合成”按钮
* 合成成功后将产出香水道具，加入背包或显示在界面上

**你应该看到的结果：**

* 玩家点两种材料，合成香水按钮变亮
* 点击后提示“成功合成 xxx 香水”
* 香水显示在香水背包区域

**✅ 第 4 步：顾客系统**

**你需要完成：**

* 每隔一段时间生成一个顾客（显示在画面上）
* 顾客提出想要的香水（如提示：“我想要 xxx 香水”）
* 玩家点击香水“出售”给顾客，获得金币

**你应该看到的结果：**

* 一个顾客小人走进来，头上冒泡泡：“我想要香水A”
* 玩家选择正确香水，点击“售出”
* 提示“交易成功！获得金币 +10”

**✅ 第 5 步：游戏循环系统 & UI**

**你需要完成：**

* 时间推进（一天多少小时）
* 玩家体力或金币系统
* 音效、背景音乐
* 游戏结束判定（如：三天没赚钱就 Game Over）

**✅ 第 6 步（可选）：优化与跨平台**

**你可以：**

* 使用 Phaser 的打包工具输出 HTML5 版本
* 加载屏幕、资源管理
* 添加动画（合成动画、顾客进出动画）
* 使用 LLM/AI 工具生成部分图标、香水瓶图案等

**功能模块分解**

**1. 香水类 Perfume**

* 属性：name（香水名称）、ingredients（香水成分）、price（香水价格）、stock（库存）。
* 方法：createPerfume（根据成分创建香水）、updateStock（更新库存）。

**2. 商店类 Store**

* 属性：inventory（香水库存）、money（玩家余额）。
* 方法：buyPerfume（购买香水）、sellPerfume（出售香水给NPC）、unlockPerfume（解锁新香水）。

**3. NPC系统**

* 属性：name（NPC名称）、preferredPerfume（NPC喜爱的香水）。
* 方法：makePurchase（模拟购买香水）。

**4. 合成系统**

* 玩家可以选择不同成分组合香水，合成后香水价格和种类不同。

**5. 界面交互**

* 显示商店中所有香水的列表、价格和库存。
* 玩家点击按钮购买香水，点击按钮合成香水。

**6. 香水解锁机制**

* 根据销售数量、时间等解锁新的香水配方。
* 每解锁一个新香水，玩家的利润和库存会增加。

**游戏设计简略步骤**

**Day 1：游戏框架搭建**

1. **初始化游戏框架**
   * 使用 TypeScript 和 Phaser 或其他2D游戏框架来初始化项目结构。
   * 设置基本的文件夹结构：src/game（游戏逻辑）、src/web（前端展示）、assets（游戏资源）。
2. **店铺管理系统**
   * 创建 Store 类，用来管理香水库存、价格和玩家余额。
   * 创建 Perfume 类，表示香水种类，包含香水名称、成分、价格等信息。
3. **香水解锁系统**
   * 游戏开始时，玩家只有一种香水（例如“玫瑰香水”）。
   * 随着时间的推移，玩家可以解锁新的香水种类（例如通过任务、销售或时间机制）。
4. **NPC购买系统**
   * 创建一个基础的 NPC 系统。NPC 会随机选择一种香水并购买。
   * 购买后，减少玩家的香水库存并增加金币。

**Day 2：实现互动和功能**

1. **香水合成系统**
   * 玩家可以合成不同的香水，例如“薰衣草 + 玫瑰”。
   * 每种组合可以通过配方解锁，合成后生成新的香水并加入库存。
2. **商店界面**
   * 创建商店界面，展示可购买香水的列表，价格以及库存数量。
   * 显示玩家余额，购买香水时更新库存和余额。
   * 实现一个按钮用于合成香水。
3. **NPC购买系统**
   * 创建一个 NPC 随机购买香水的机制：根据库存数量和价格，NPC 会定期进入商店购买香水。
   * 玩家可以看到 NPC 的购买行为，增加互动性。
4. **升级和解锁新香水**
   * 创建一个系统，让玩家完成任务、达到一定销售量或时间进度后解锁新香水配方。
   * 新香水可以带来更高的利润或增加玩家的声望。

**Day 3：增强游戏体验**

1. **美化界面与动画**
   * 为香水添加图标或图像，以视觉化每种香水。
   * 添加动画效果，如购买时的闪烁、香水合成时的过渡动画等。
2. **背景音乐和音效**
   * 为商店界面和购买过程添加音效（如铃声、合成时的魔法音效等）。
   * 可以添加背景音乐，增强游戏氛围。
3. **游戏存档与成就系统**
   * 添加存档系统，玩家可以保存游戏进度。
   * 设计简单的成就系统，例如“售出100瓶香水”解锁一个特别配方。

**Day 4（可选）：优化与部署**

1. **优化性能**
   * 确保游戏在不同设备上流畅运行，优化性能，减少内存消耗。
2. **打包与发布**
   * 使用 http-server 部署 Web 版本，确保在浏览器中正常运行。
   * 如果需要，可以考虑使用 Electron 打包成桌面应用。

**Day 1: 基础系统**

* 搭建游戏结构和开发环境（你已经完成）
* 实现香水类（Perfume）和商店类（Store）
* 显示“Rose Perfume”，价格、库存
* 实现金币系统，初始给玩家 100 gold
* 显示 UI：香水、库存、金币（先用文本表示）

**🟨 Day 2: 交互与 NPC**

* 每隔几秒生成一个 NPC，尝试购买香水
* NPC 购买逻辑：如果有库存则交易，金币增加
* 添加合成香水按钮（不必真的合成，模拟添加库存即可）
* 增加 UI 提示：谁来买了什么，库存变了

**🟦 Day 3: polish + 打包**

* 美化 UI：简单图片或颜色区分不同香水
* 加一个 loading/开始界面（可选）
* 进行性能测试、构建 Web 版本
* 打包为 Windows 可执行程序（使用 Electron 可选）

Game Title:

Perfume Shop Tycoon

Game Concept:

Perfume Shop Tycoon is a casual 2D business simulation game where players run their own perfume shop. They will craft and sell perfumes by mixing various ingredients, manage their store, and earn money to expand the business.

1. Game Overview

Genre:

Simulation, Tycoon, Casual

Platform:

Web (Browser) and Windows (Desktop)

Target Audience:

Casual gamers, simulation game lovers, fans of management and crafting games.

Core Gameplay:

Players start with a small perfume shop where they can craft perfumes by combining ingredients.

The goal is to earn money by selling perfumes to customers and use the profits to upgrade the shop and acquire new ingredients.

Players can experiment with different ingredient combinations to create unique perfumes that are more popular or expensive.

The game includes time management, inventory management, and customer satisfaction mechanics.

2. Game Mechanics

Perfume Crafting:

Players combine various ingredients (e.g., lavender, rose, vanilla) to create perfumes.

Each perfume has a unique set of ingredients that affect its price and popularity.

The price of the perfume depends on the number of ingredients used.

Shop Management:

Players can set the price of each perfume and adjust inventory based on demand.

Players need to balance supply and demand to keep customers happy and make a profit.

Profits can be used to buy new ingredients, expand the shop, or upgrade equipment (e.g., better perfume bottles, new marketing tools).

Money System:

Players start with a set amount of money to buy their first ingredients and create a small batch of perfume.

As players sell perfumes, they earn money that can be reinvested into their shop.

If players run out of money, they need to carefully manage their resources and sales to continue.

Time System:

Perfumes take time to craft, and there is a limited number of orders the shop can process in a day.

Players can speed up crafting or set schedules to optimize production.

3. Visual Style and Art

Art Style:

Simple 2D sprites for ingredients, bottles, and perfume bottles.

Clean, minimalistic UI design with a focus on ease of navigation.

Soft pastel colors to evoke a calming, elegant atmosphere (e.g., light purples, pinks, and blues).

Assets Needed:

Ingredient icons (e.g., Lavender, Rose, Vanilla)

Perfume bottle designs

Shop and store assets (counter, shelves, etc.)

Customer avatars (optional)

Background art (e.g., shop interior)

Simple animations for perfume crafting (e.g., swirling liquid, steam)

4. User Interface (UI)

Main Screen:

Shop view: Display the perfume bottles, crafting station, customer interaction button.

Shop inventory: List of available ingredients and perfumes with prices.

Money display: Show current funds and any ongoing sales.

Crafting Screen:

Display the available ingredients and a “mix” button to craft perfumes.

Show a progress bar for crafting time.

Display the name and price of the perfume being created.

Customer Interaction:

Customers will come into the shop, browse perfumes, and make purchases based on their preferences (which could depend on ingredient combinations or prices).

Display the number of customers waiting, total sales, and time of day.

5. Sound and Music

Music:

Light, relaxing background music to enhance the atmosphere (e.g., soft instrumental music).

Sound Effects:

Clicking sounds for button presses.

Crafting sounds for mixing ingredients.

Cash register sounds when customers make purchases.

Background ambiance sounds (e.g., gentle shop noises).

6. Technical Requirements

Technology Stack:

Game Engine: Phaser.js (2D game engine for web-based games).

Programming Language: TypeScript (to ensure strong typing and scalability).

Build Tool: Webpack or other bundlers to handle assets and code.

Deployment: HTML5 for browser support, and Electron (optional) for the Windows desktop version.

Performance Requirements:

The game should run smoothly in modern browsers.

Fast loading times for assets (images, sounds).

Smooth animation and transitions for gameplay.