

# 项目报告

张子墨，李欣桐

2025年1月8日

## 1 引言

本项目为 SI100B 课程的最终项目，项目文件已上传到 <https://github.com/Elena051006/Harvest-Moonlight>。我们开发了一款名为 Harvest Moonlight 的种菜小游戏。该游戏旨在为玩家提供一个既充满田园诗意又充满紧张感的游戏体验。在游戏中，玩家将化身为一名农夫，亲身体验播种的希望与收获的喜悦。

Harvest Moonlight 的核心玩法围绕着农夫的日常活动展开，包括种植作物、收获果实、管理资源等。游戏中的紧张感来自于捣蛋鬼兔孩子的随机出现，它会在玩家周围随机出现并定点追踪玩家的位置，试图干扰玩家的农耕生活。这种设计不仅增添了游戏的趣味性，还考验了玩家的操作技巧和应变能力。

本报告将详细介绍 Harvest Moonlight 的游戏设计、实现过程以及创新点。

## 2 项目实施

### 2.1 场景

#### 2.1.1 场景布局概览

在游戏中，我们精心设计了 5 个核心场景，为玩家提供丰富多样的

游戏体验。

主场景——农场（见图 1）：这里是玩家的主要活动区域，其中包括了农田、小屋、商店等关键区域，玩家将在这里耕耘土地、种植作物，体验田园生活的乐趣。

初始场景（见图 2 和图 3）：作为游戏的起点，玩家在此可以详细了解游戏规则、操作指南以及背景故事，为即将展开的农场生活做好准备。

游戏结束场景（见图 4 和图 5）：当玩家的农场之旅告一段落时，将进入此场景。

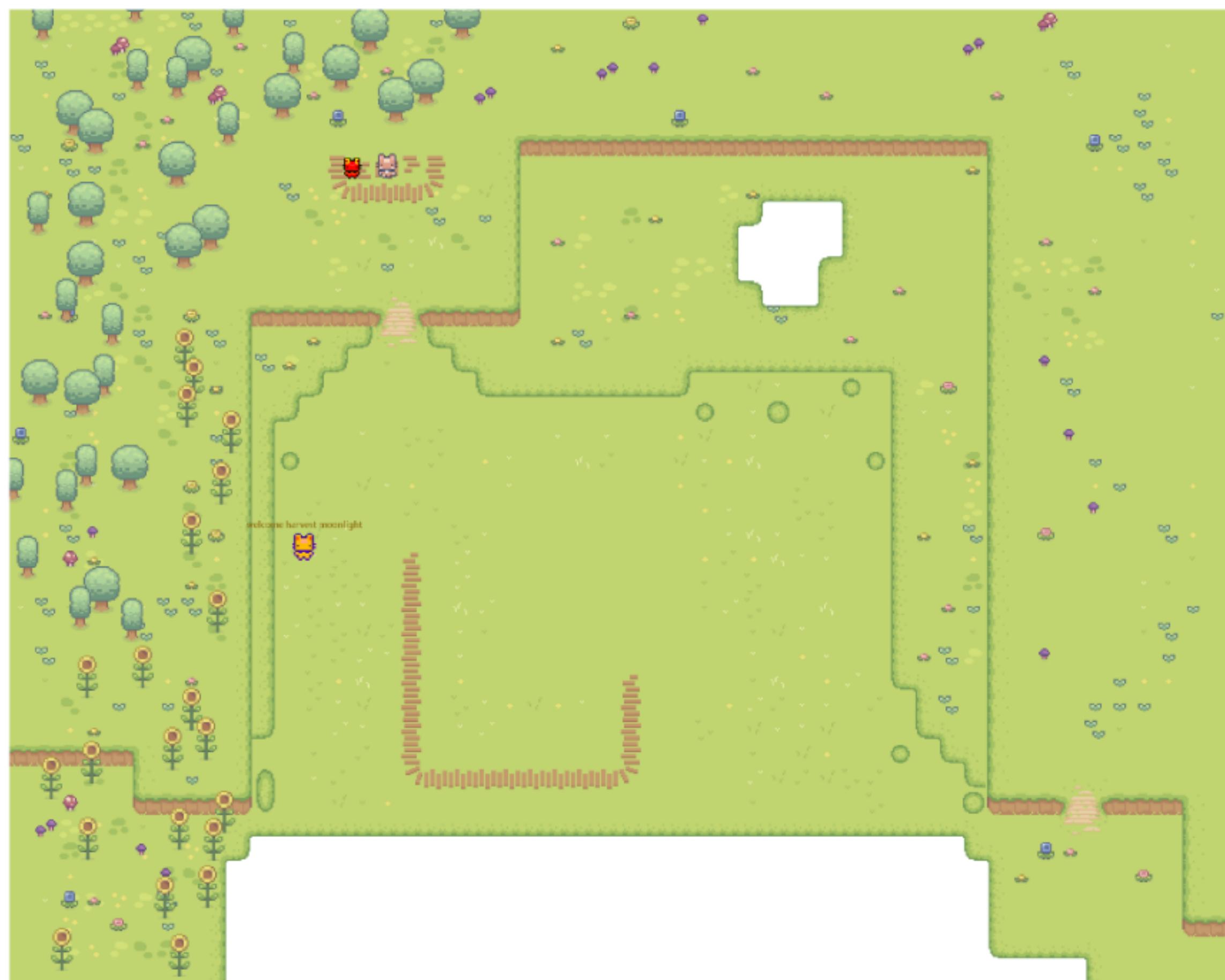


图 1：主场景



图 2：游戏开始

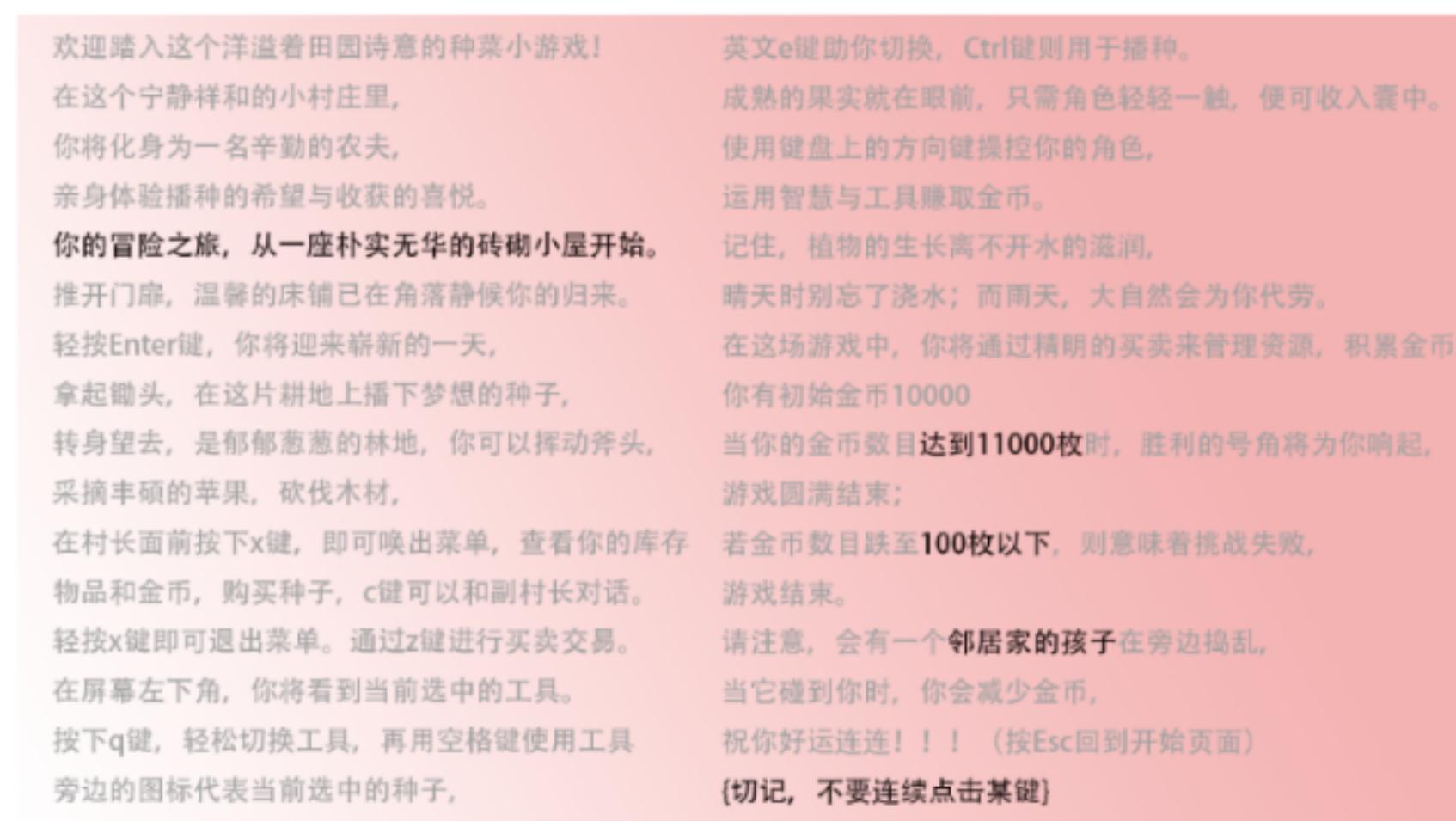


图 3：游戏说明



图 4：游戏失败



图 5：游戏成功

### 2.1.2 核心场景布局

游戏中包含一个主要场景，详见图 1。

在这个主场景中，玩家将体验到多样化的游戏活动。砖砌的小屋内提供了角色休息的场所；小屋门外是一片耕地，角色可以在这里使用锄头和浇水壶等工具进行农作；耕地的外围是郁郁葱葱的林地，角色可以手持斧头在这里采摘苹果或砍伐树木；小屋的北面坐落着村长的小屋，在其门前，角色可以唤出菜单界面，与店主进行商品交易，同时也可以与村长进行交流。

我们的场景以广阔的草地为基础，在此基础上，我们精心搭建了三维设计元素。利用树木、墙壁等障碍物精灵构成了场景的边界，同时构建了小屋、河流以及向日葵等装饰性精灵，共同构成了一个完整的游戏世界。这些背景精灵按照一定的顺序进行渲染，为玩家呈现出一个层次分明、生动立体的游戏环境。

在主场景中，我们设置了游戏镜头跟随机制，在 level.py 文

件中，我们通过定义 CameraGroup 类并将其集成到 Level 类中，巧妙地实现了游戏摄像机的动态跟随功能。CameraGroup 类继承自 Pygame 的 pygame.sprite.Group，并对精灵组的绘制方法进行了重写。该方法通过精确计算玩家角色位置与屏幕中心的偏移量，实时调整场景中所有精灵的绘制坐标，确保它们在摄像机的新视角下正确渲染。这一机制确保了游戏视角始终紧随玩家角色，打造出无缝衔接的视觉跟随效果，极大地增强了玩家的沉浸感和整体的游戏体验。

### 2.1.3 互动物品设计

我们的游戏中有 12 种基本的互动物品，如图 6。

在游戏中，围栏、树木、林地和墙壁等元素均充当了障碍物的角色，限制了角色的行动路径，确保角色无法穿越这些固态障碍。同时，湖水、家具、花草树木等元素则用于装饰场景，同样采用三维设计，使得角色无法穿行其中。在有雨滴的情况下，湖水会展现出动效效果，而土地在雨水作用下也会呈现出相应的动态变化。这些互动元素在地图砖块渲染之后进行绘制，从而营造出它们位于地图砖块之上的视觉效果。游戏中部分互动元素的特定功能将在后续的游戏机制部分进行详细说明。



图 6: 互动物品

主界面的左下角是工具栏的表现，工具可以根据按键进行调整，具体实现请见 `over\ay.py`。



图 7: 工具栏

## 2.2 角色

### 2.2.1 主要角色

在游戏中，玩家可通过上、下、左、右四个方向键来操控角色在游戏世界中的移动。在英文输入法下，按下 Q 键可切换不同的工

具，空格键用于使用当前选中的工具，E 键用于切换种子种类，而 Ctrl 键则用于播种操作。

游戏开始时，主角默认出现在家门口。玩家可以通过键盘输入来控制角色进行水平或垂直方向的移动，详细控制逻辑可在 player.py 文件中的 import\_assets 函数和 move 函数中查看。

### 2.2.2 普通 NPC

果实精灵（如图 8）：在播种并浇水之后，植物将随着时间的推移逐渐成长。当果实成熟时，若角色靠近，果实将发光以示收获成功，收获的同时，果实数量将增加，并播放“success”音效以庆祝收获的喜悦。具体的收获逻辑和音效播放可在 level.py 文件中的 player\_add 函数中查看。果实收获采用了碰撞检测系统，允许角色直接通过接触来收集成熟的果实。这一碰撞系统的具体实现细节，可以在 level.py 文件中的 plant\_collision 函数中找到。

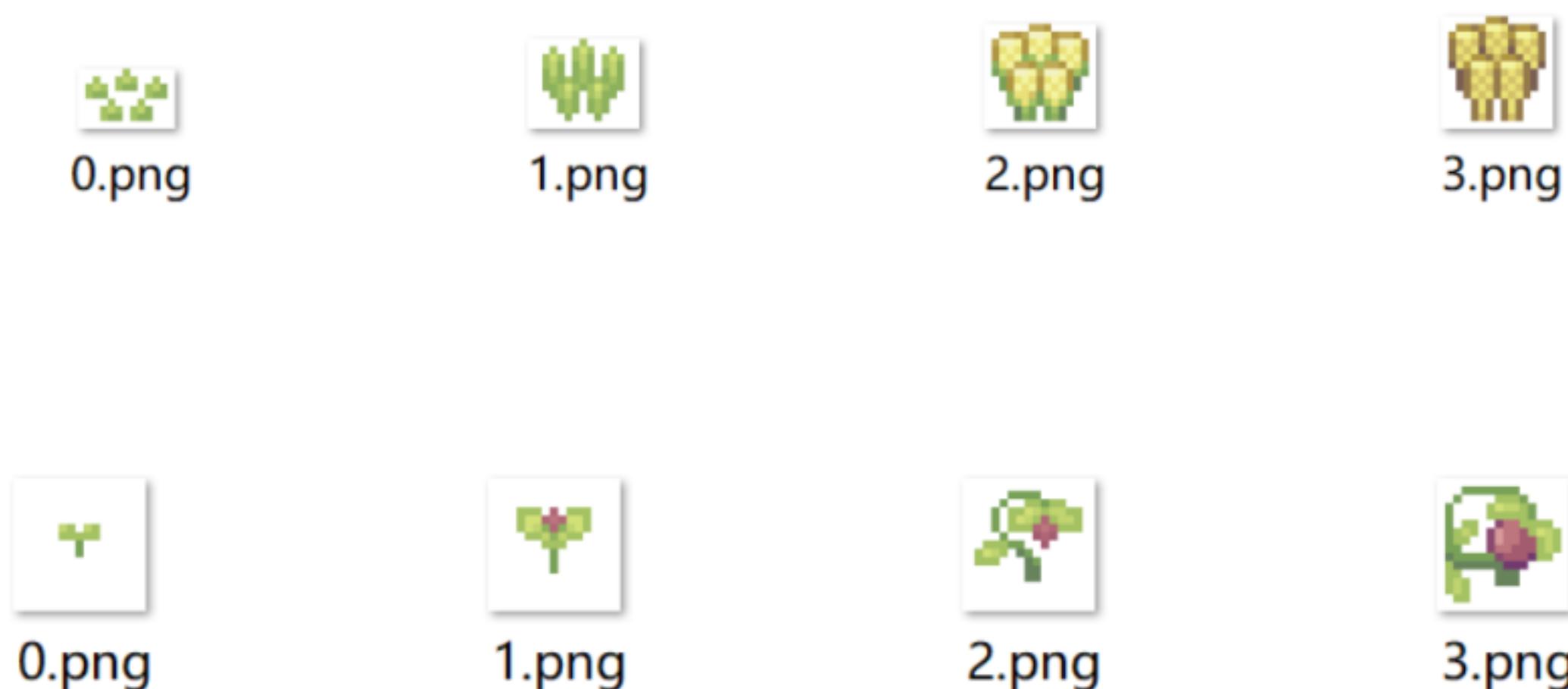


图 8：果实

树木精灵（图 9）：树木以三维形态存在，不仅美化了场景，还作为障碍物界定了林地的边界。树木分为两种类型：大树和小树。当树木被砍伐后，将以树墩的形式保留在原地。在游戏进行过程中，除非游戏重启，否则这些树墩不会重新长出树木，从而为游戏世界增添了持久性和现实感，保持系统详细请看 level.py 中的 reset 函数。

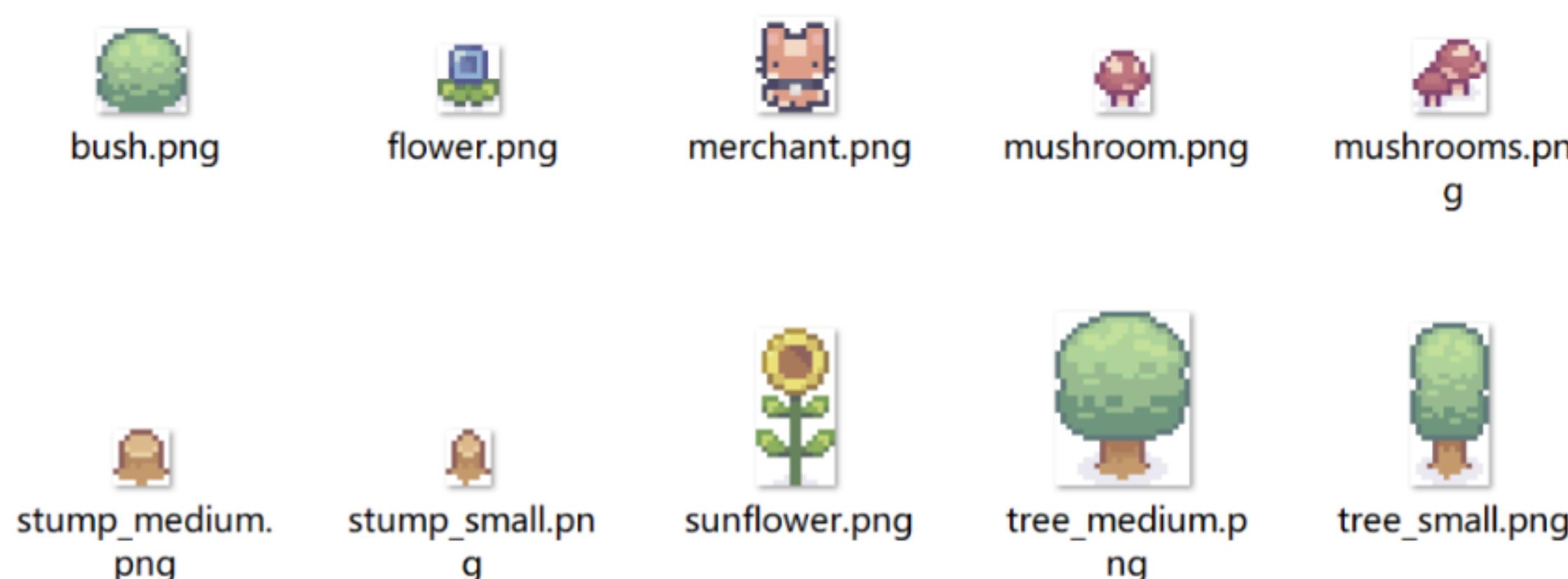


图 9：树木

苹果精灵：在每一轮游戏中，苹果的出现是随机的。如果在本轮游戏中树木上长出了苹果，玩家可以使用斧头工具进行采摘。一旦苹果被采摘，它们将不会在当前游戏中重新出现。这一机制的维护和重置逻辑可以在 level.py 文件中的 reset 函数中找到。

雨点精灵（图 10）：游戏中的天气系统由 AI 智能调控，其决策依据是玩家当前所持有的金币数量。具体的决策逻辑可以在 llm\_agent.py 文件中的 decide 函数中查看。雨点精灵分为三种不同的类型：长、中、短，它们随机地在屏幕上生成。当雨点落在湖

水和土壤上时，会产生动态效果，包括大、中、小三种规模的溅起水花，为游戏场景增添生动的气象变化。



图 10：雨滴和水坑

### 2.2.3 友好 NPC

在村长的小屋中，玩家将遇到两位友好的 NPC：一位是村长，激活菜单后，玩家可以与村长进行交易；另一位是副村长，激活菜单后，玩家可以与副村长进行对话。请注意，与副村长的交流需在英文模式下进行。

副村长的对话功能采用了 LLM (Language Learning Model) 技术，具体的对话处理逻辑可以在 `llm_agent.py` 文件中的 `chat` 函数中找到。

### 2.2.4 捣蛋敌人

灰色的兔小孩是角色邻居家的一个调皮角色。它以捣蛋鬼的形象出现，当它围绕在角色身边时，玩家需要给予它金币以避免它的哭闹。因此，一旦捣蛋鬼灰色兔小孩进入角色周围一单位长度的范围内，系统将会从角色账户中扣除一定数量的金币。

## 2.3 游戏机制

### 2.3.1 核心机制

在宁静的小村庄中，玩家通过操控角色与土地、林地、邻居孩子和成熟果实进行互动，耕作、砍伐、支付金币和收获果实，以管理农场和资源。通过每日的键操作来开始新的一天、移动、切换工具和种子、交易和管理库存，最终目标是积累至 11000 枚金币以赢得游戏，或避免金币跌至 100 枚以下以免挑战失败，整个游戏体验以金币数量作为成绩衡量。

### 2.3.2 碰撞系统

我们为角色和游戏对象创建一个名为 `hitbox` 的矩形碰撞检测区域，来实现碰撞检测。我么将所有需要进行碰撞检测的角色和精灵添加到 `collision_sprites` 组中，在游戏循环的每个更新步骤中，系统会遍历 `collision_sprites` 组，检查角色与其他对象之间的碰撞。如果检测到碰撞，系统将根据碰撞的类型（如与植物、地形或其他角色）来调整角色的位置或执行相应的动作，如收获植物。

### 2.3.3 资源系统

我们将资源查看安排在村长小屋的菜单中，当菜单被激活时，主场景的动态将暂时停止。我们对游戏菜单进行了初始化设置，其中包括定义菜单的背景颜色和默认字体，并指定了触发菜单显示的按键以及打开对话框的按键，设定了角色目前拥有的资源的数目，并允许角色和村长进行贸易交互，允许角色在菜单中进行交易和与

AI 进行对话。相关的实现细节可以在 `menu.py` 文件中找到。菜单内容的初始化过程则在 `player.py` 文件中展示。

## 2.4 LLM 智能体系统

LLM 智能体系统被集成到游戏中，以提供动态的 NPC 对话和决策支持。我们在 `llm_agent.py` 中定义了两个基于语言模型的代理类，`LLMChatAgent` 和 `LLMDecisionAgent`，均继承自 `LLMAgent` 基类，两个代理类都通过 OpenAI 客户端与语言模型进行通信。在文件的末尾，我们设置了一个打印语句，表示 `LLMDecisionAgent` 已成功加载。

### 2.4.1 对话系统

`LLMChatAgent` 继承自 `LLMAgent`。在初始化方法中，我们描述了游戏中的角色和规则，例如主角、商人和捣蛋鬼的角色和动作，作为 AI 的基础信息。该类包含一个 `send_message` 方法，该方法允许发送用户消息并获取助手回复。它将用户消息添加到消息列表中，并通过 OpenAI 客户端发送消息以获取回复。然后，它将助手的回复添加到消息列表中并返回。

### 2.4.2 决策系统

`LLMDecisionAgent` 也继承自 `LLMAgent`。在初始化方法中，我们描述了基于输入数字奇偶性的决策逻辑。该类包含一个 `decide` 方法，该方法接受一个整数值（表示金钱），发送这个值作为消息，并通过 OpenAI 客户端获取回复。然后，它根据回复的值来决定是

否下雨（raining）。

## 2.5 游戏性

### 2.5.1 主菜单

我们的游戏有三种界面，主界面、失败界面与胜利界面。

进入游戏时会显示主菜单，此时玩家可以按下空格键查看游戏说明，按下 ENTER 键以进入游戏。当游戏胜利时，会出现成功界面；当游戏失败时，会出现失败界面，按下 ENTER 键可重新开始游戏。

### 2.5.2 BGM

我们的游戏在主场景有一个 BGM，根据主场景的天气状况，会叠加天气音乐。晴天时，会播放鸟叫和流水声；雨天时，会播放雨声和风声。当玩家成功收获苹果、砍下树木或者收获果实的时候，会播放收获成功的 BGM。

## 2.6 代码

仓库中 Game 为游戏所在文件，用编辑器打开后，调整路径运行 Main.py，或直接通过在 Gam 目录打开终端，调整路径并输入 python Main.py 运行，即可进入游戏。

程序采用面向对象编程的编程方式，在代码中，我们通过继承和封装，保证各个功能能够独立运行互不干扰，同时减少重复的代码，使代码更加简洁明了，并且代码中难以理解的地方我们也全都

添加了注释。

下面罗列程序中的各个文件的相应作用：

- 1、`level.py` 文件是游戏的关键模块，负责构建和管理游戏关卡，包括地图加载、角色创建（玩家和敌人）、环境元素（如树木和水）、界面显示（如天气效果和商店菜单）以及音效播放，同时集成 AI 决策代理，实现了游戏循环中的主要逻辑和摄像机跟随功能。
- 2、`llm_agent.py` 实现了大语言模型的运用，包含 `LLMAgent` 基类，两个子类：`LLMChatAgent` 和 `LLMDecisionAgent`。  
`LLMChatAgent` 用于发送和接收聊天消息，而 `LLMDecisionAgent` 则根据给定的金币数量决定是否下雨。
- 3、`main.py` 是游戏的主程序，定义了 `StartScreen` 类，负责游戏的开始屏幕显示、结束屏幕显示以及主界面运行。
- 4、`menu.py` 定义了 `Menu` 类，用于游戏中的商店界面交互，包括物品的购买与出售、聊天功能以及用户界面的显示逻辑。这个类初始化时接收一个玩家对象，并设置了显示表面、字体、菜单选项、颜色和聊天代理。它提供了展示和隐藏商店、显示玩家金币、处理商店导航和事件、以及聊天输入和显示的功能。
- 5、`overlay.py` 定义了 `Overlay` 类，用于在游戏界面上添加并显示玩家当前选择的工具和种子图标的覆盖层。

6、player.py 定义了 Player 类，用于表示和控制游戏中的玩家角色。这个类处理了玩家的初始化、动画、输入、工具和使用工具、获取目标位置、使用种子、导入资源、动画处理、输入处理、碰撞检测、移动和交互检查以及游戏逻辑更新等等。

7、settings.py 设置了游戏中基本精灵的基础数据，如村长小屋中各个商品的价格，果实生长的速度等等。

8、sky.py 中定义了 Sky 类，它涉及显示和更新天空元素，同时创建基础草地和湖面在雨滴的作用下的动态效果。

9、sprites.py 定义了 Generic 类，初始化、动画处理、更新方法、伤害处理、死亡状态检查以及创建水果等。

10、settings.py 提供了在游戏中导入和管理图像资源的功能。

11、transition.py 定义了 Transition 类，利用 pygame 库实现屏幕过渡效果，通过颜色渐变来控制过渡，呈现昼夜更替的视觉效果，并在颜色值达到特定阈值时触发重置或更新玩家状态，即实现开启新的一天的动画效果。

12、timer.py 定义了 Timer 类，允许用户设置计时器持续时间，激活或停用计时器，并在计时器结束时执行一个可选的函数。

### 3 创意

### 3.1 天气系统的闭环设置

在项目中，我们部署了决策系统，以实现对游戏内天气状况的动态调控，从而提升游戏的动态性和玩家的互动体验。

在天空视觉效果的设计上，我们精心调配了天空的色调，并开发了相应的动态效果算法，确保了晴朗天空的稳定性以及雨天雨滴效果的随机性。雨滴与不同表面的交互效果，以及自然灌溉系统的实现，均在 `sky.py` 文件中详细记录。

此外，背景音乐的同步变化机制也得到了精心设计，以匹配不同的天气场景。在晴朗天气中，我们叠加了鸟鸣声效；在雨天，则增加了细雨的音效，以营造更为丰富和沉浸的听觉环境。相关的实现细节和配置均在 `level.py` 文件中详细说明，以确保音效与游戏场景的高度协调。

### 3.2 动态追踪机制设计

我们实施了定点追踪机制，旨在实现捣蛋鬼角色对玩家位置的实时监控，从而提升游戏的紧张氛围和互动性。该机制通过连续监测玩家角色的地理位置，并据此动态调整捣蛋鬼的行动轨迹。当捣蛋鬼进入以玩家为中心、半径为一个单位长度的圆形警戒区域时，将触发惩罚事件，导致玩家损失一定数量的金币。因此，玩家需持续关注并避免与捣蛋鬼的接触。具体的实现细节可在 `level.py` 文件中的 `set_player_pos` 函数中查阅。

为了增加游戏的不可预测性，捣蛋鬼的初始位置被设定为在玩家

家角色周围 30 个单位长度的范围内随机生成。这一设计使得捣蛋鬼成为游戏中一个不可预测的变量，要求玩家在从事耕作和收获活动的同时，保持对捣蛋鬼动向的警惕，以免遭受金币损失。此外，玩家可通过特定的策略暂时限制捣蛋鬼的行动。此类游戏设定不仅考验玩家的操作技能，同时也考验了其观察力和策略规划能力。

### 3.3 动态效果的创新应用

游戏中的角色和背景已全面采用动态效果，以增强玩家的视觉体验。

在角色设计方面，我们利用 Pygame 的功能为角色静止时增添了耳朵动态效果，增强了角色的活力。角色根据不同朝向展现出相应的姿态，确保了动作的自然流畅。移动时，角色执行包括抬腿等一系列动作，模拟真实行走，且移动速度的变化也体现在腿部的运动上，增强了游戏的真实感。手持工具时的不同造型和动作均在 player.py 的 import\_assets 函数中详细定义。工具使用动画的加入，使得操作更加生动逼真。

天气效果方面，我们实现了雨滴动态效果，增强了游戏的视觉吸引力和沉浸感。雨滴随机分布、不同大小和形态，以及落在不同表面的效果，如随机生成的水坑大小和位置，都增加了场景的自然多样性。雨天时，耕地自动播放浇水动画，丰富了游戏体验并贴近现实。

此外，我们构建了动态昼夜循环系统，通过屏幕基准色调和过

渡层的颜色变换，模拟了从黎明到深夜的光影变化，详细实现可见 transition.py 文件。

游戏还设有开启新一天的按键，展现了从夜晚到黎明的动态效果。新的一天开始后，浇水的果实会按设定速度生长，而被砍伐的树木和摘下的苹果则不会重生，为游戏世界增添了现实感。

### 3.4 场景生动性体验

我们的游戏场景主要采用三维设计，致力于在图层上打造出逼真的视觉效果。从树木、向日葵、小花小草到蘑菇等装饰性元素，均以三维形态呈现，实现了与角色的碰撞交互，增强了游戏的沉浸感。

在土壤设计方面，我们精心构建了一个二维网格系统，用以标识游戏中可耕种的区域。每个可耕种的单元格都对应一个碰撞检测矩形，这些矩形用于精确认识玩家何时进行点击或与土壤互动。基础草地采用了三维视觉效果，允许玩家使用锄头工具进行开垦。开垦过程中，草地的颜色从绿色转变为棕色，并在色块边缘添加阴影效果，增强了视觉立体感。耕地浇水后，颜色进一步加深，并模拟了浇水动画效果。我们设计了多种不同方位的开垦土壤砖块状态，根据单元格周围环境状态确定土壤砖块类型，并选择相应的图像进行展示，确保了土壤层的视觉丰富性和与游戏逻辑的紧密联系。通过这种细致的土壤砖块类型设计，我们能够创造出更加丰富和逼真的地形效果，为玩家带来身临其境的游戏体验。相关的实现细节请

参考 soil.py 文件。

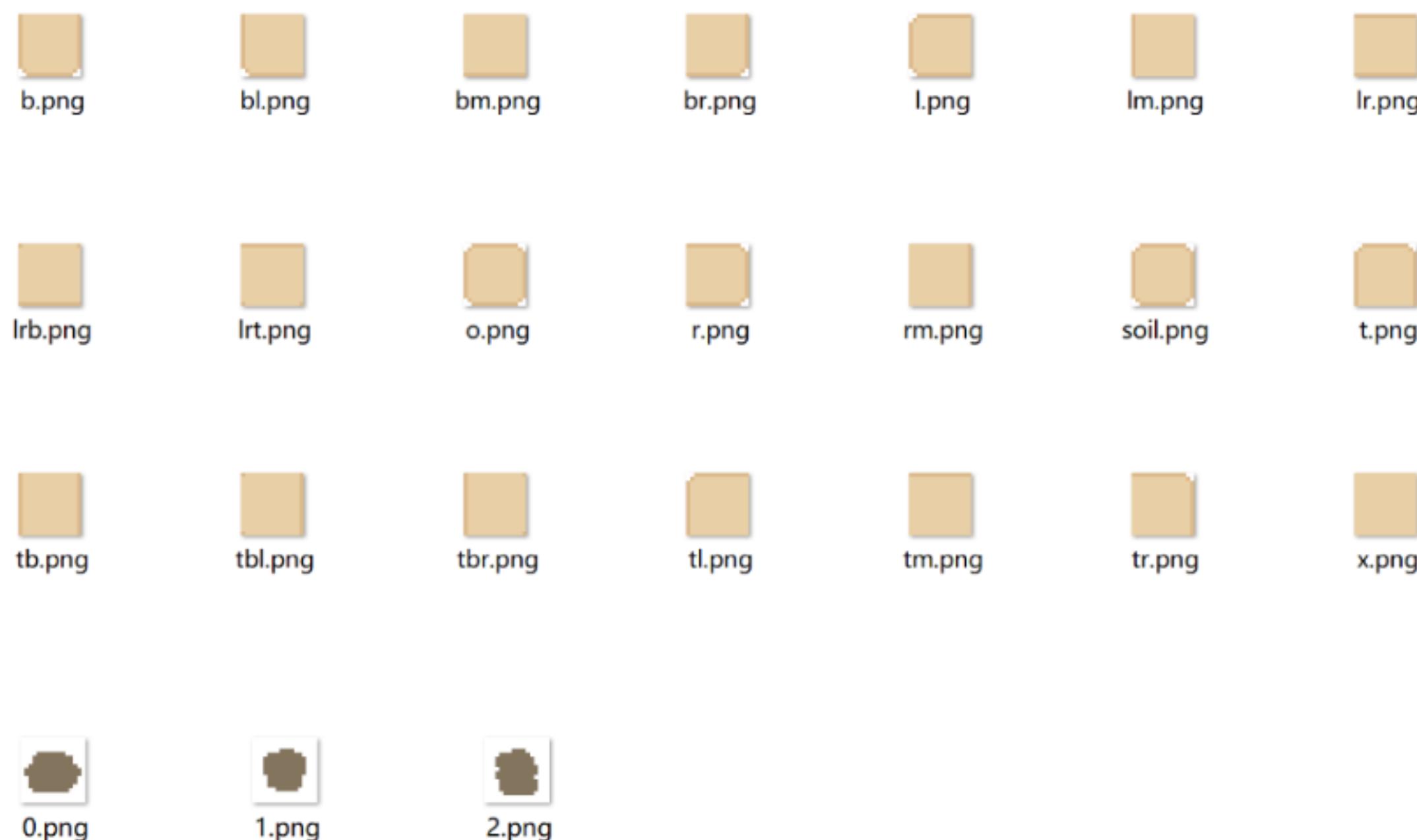


图 11：土壤的各种类型

### 3.5 定点感应

我们精心设计了一套定点感应机制，确保玩家仅在特定场景和精确方向下，通过按键操作才能激活特定的机关。例如，玩家需站在卧室前的地毯上，并且面向左侧时，按下‘R’键，才能启动新一天的序幕。

### 3.6 维护场景整体风格的和谐统一

我们精心挑选并采用了统一的田园风格素材，小到小花蘑菇，大到小路篱笆，都弥漫着浓厚的田园气息。与此同时，BGM 选择了柔和而舒缓的轻音乐，再叠加略微弱化的自然声，为玩家带来一种心灵上的宁静与放松，仿佛置身于真正的田园诗画之中。这样的设计，不仅增强了游戏的美感，也让玩家在探索和互动的过程中，感

受到一种平和与自然的融合。