# 项目报告

王子睿、胡俞中、段林沛

2025 年 1 月 12 日

# 目录



# 1 简介

在项目根目录下输入 `make` 或者 `python src/main.py` 可以运行游戏。

本项目为 SI100B 课程的最终项目，项目文件 已上传到了 GitHub。我们制作了一款类银河战士恶魔城游戏，玩家陷入梦境之中，手握一把丝线之剑，可在平台间跳跃与怪物搏斗。

战斗采用即时制，玩家通过键盘控制来进行移动和攻击，躲避怪物攻击的同时对其造成伤害。玩家击败怪物之后可以获得灵魂货币，到商人 NPC 处可以用灵魂货币购买升级，击败怪物一定次数之后可以获得胜利。

程序的入口是 `main.py`，其中的 `Game` 类负责游戏的整体逻辑。游戏的各个部分被分割到不同的模块。

- `settings.py`：存储各种可调参数
- `level.py`：地图的控制
- `tile.py`：地图中的障碍物
- `player.py`：玩家的控制
- `weapon.py`：玩家的三种武器
- `npc.py`：NPC 的控制
- `enemy.py`：敌人的控制
- `menu.py`：游戏的开始与结束画面
- `keys.py`：处理玩家输入
- `utils.py`：一些工具函数
- `debug.py`：一些调试用的函数

# 2 项目内容

## 2.1 场景

### 2.1.1 大型场景

游戏的场景是一个大型地图，地图中有两位 NPC 与会复活的敌人。

图 1: 地图

整个地图由 `level_dream.py` 中的 `LevelDream` 类控制，它继承自 `level.py` 中定义的 `Level` 类。地图中的障碍物、敌人、角色与 NPC 都是在这里被创建实例，并在游戏循环中被更新。

游戏中的图像可能会重叠，为了让它们被正确显示我们定义了 `VisibleGroups` 类。先对可见图像按照 y 坐标排序，然后按照” 物品”、“敌人与 NPC”、“主角” 的先后顺序绘制。

### 2.1.2 交互对象

地图中的障碍物在 `tile.py` 中定义。每个障碍物都有自己的碰撞箱，玩家与会与之发生碰撞。

图 2: 地图中的障碍物

## 2.2 角色

### 2.2.1 主角

**属性** 玩家拥有生命值与魔力值，被敌人打到会损失生命，释放魔法会消耗魔力。

**移动** 玩家可以按下 <A> 和 <D> 来左右移动，按下 <Space> 跳跃，按下 <Shift> 冲刺，按住 <Shift> 疾跑。为了让玩家的移动更加自然，我们在玩家的跳跃过程中引入了重力加速度，并在检测到玩家落地的时候应用摩擦力。

**攻击** 玩家可以按下 <J> 造成普通攻击，按下 <H> 丢出武器，进行远程攻击，按下 <U> 挥舞丝线，释放魔法。


图 3: 玩家属性显示

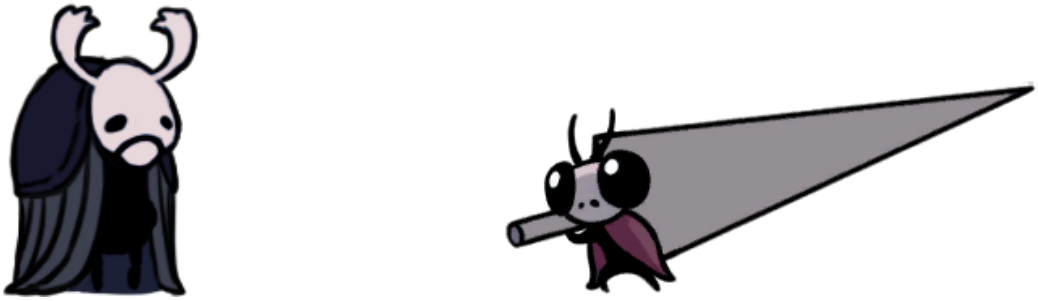
### 2.2.2 NPC

游戏中有两位 NPC，一个是向导，另一个是铁匠，他们都继承自 `npc.py` 中定义的 `NPC` 类。向导会在游戏开始时向玩家介绍游戏的基本操作，铁匠可以升级玩家的武器。

当玩家接近 NPC 的时候按下 <R> 可以开始对话，如果需要回复可以按下 </> 后输入内容，按下 <Enter> 发送。玩家打开与铁匠的对话后按下 <B> 可以升级武器（消耗 10 单位货币）。

为了不漏掉玩家在输入对话的时候按下的每一个按键，我们在 `keys.py` 中通过处理游戏中获取到的每一个 `pygame.event.Event` 事件来管理输入。

具体的对话内容我们调用了 `python-openai` 库的 API，其中可能会有一些不可打印字符，我们在 `utils.py` 使用正则来“标准化”了一遍返回的字符串。



(a) 向导　　(b) 铁匠

图 4: NPC

### 2.2.3 敌人

地图中存在可以复活三次的敌人。敌人在每一帧都有一定概率转向，在距离初始位置太远时会强制转向。敌人每次被击败都会展示死亡动画，一段时间后复活。

初始时候玩家需要三次普通攻击才能杀死一个敌人，但是在升级两次之后就只需要一次攻击。每杀死一次敌人都会获得 5 单位货币。

(a) 移动　　(b) 死亡

图 5: 敌人

## 2.3 游戏机制

**核心机制**　玩家可以自由探索地图，与 NPC 对话，与敌人战斗。游戏的胜利条件是击败 `WIN_COUNT` 次敌人。

**碰撞系统** 游戏每一帧中玩家都会调用 `move()` 方法，其中会先进行 x 方向的移动，完成碰撞检测，再处理 y 方向的移动，最后第二次碰撞检测。每次碰撞检测玩家会与可碰撞精灵组的每一个元素进行碰撞检测。由于碰撞箱都是矩形，两两碰撞检测可以 O(1) 完成，在目前的数据规模下并不会造成卡顿。

**资源系统** 玩家通过击败敌人获得灵魂货币，在铁匠处升级武器可以消耗货币。

图 6: 货币

## 2.4 游戏性

### 2.4.1 菜单

菜单 UI 界面通过 photoshop 制作完成的花纹、游戏名、按键均由十分精美。点击 START 图标即可进入游玩，SETTING 图标可以通过按钮来调节音量，单击 QUIT 即可退出游戏。将鼠标置于三个按钮上会使按钮变粗并发出声音，其中涉及了碰撞的判断。此外，经过研究我们引入了 SPEECH RECOGNITION 功能，这使得玩家可以通过使用语音输入来控制菜单的一切功能，将会在 **语音识别系统**部分详细说明。


图 7: 菜单

### 2.4.2 BGM

游戏在开始菜单，游戏结束有不同的背景音乐。我们使用了 `pygame.mixer.music` 来播放音乐，可以通过菜单中的按钮来调节音量。

# 3 创意

## 3.1 从零开始的游戏

本项目没有使用助教提供的模板，整个代码的的框架都是自己设计实现的。虽然这可能给我们带来了一些不必要的麻烦，但是也让我们更加了解了整个游戏的实现细节。而且能够实现更不一样的游戏机制。

## 3.2 调试模式

本项目在开始的时候就为了调试设置了 `debug.py` 文件，其中的 `display()` 函数在有些时候能够比 `print()` 更优雅的显示变量的值，而的 `FreeCamera` 类可以代替玩家测试地图。另外本项目在运行的时候可以传入 `--DEBUG` 参数，开启调试模式并显示碰撞箱。

## 3.3 语音识别系统

在开始菜单界面其实是可以通过语音控制的。我们使用了 `python-speechrecognition` 库来调用 Google Speech API 来实现这个功能。玩家可以高呼“开始”、“声音调大”、“声音调小”、“静音”、“退出”来控制菜单。

## 3.4 杂项

- 为了让玩家的移动更加自然，我们在玩家的跳跃过程中引入了重力加速度，并在检测到玩家落地的时候应用摩擦力。
- 为了让游戏在不同分辨率下自然显示，游戏中几乎每一个类都有 `scale` 参数。