项目报告

施云翀，马嘉欣，徐奕

2025 年 1 月

1 引言

本项目为 SI100B 课程的最终项目，项目文件已上传至：https://github.com/ProjectEggGame/PikyorEgg.git。

《捡蛋》是一款任务闯关类的养成游戏。玩家将作为一只被主人抛弃的小鸡醒来，通过与世界的交互，完成一系列任务，最终完成孵出小鸡、繁育后代的目标，这个游戏希望能通过沉浸式的体验让玩家感受到成长和生育的美好，用小鸡眼睛看生育，从而从某种程度上帮助我国生育率回升。

两个世界、五个任务、无数种地图、五类 NPC、超多交互实体，该游戏将带你用小鸡的眼睛重新看看这个世界。在主世界中，你需要捡拾米粒、收集树枝、通过转送门进入异世界，并且与此同时提防蠢蠢狐狸的袭击。而后，你会遇到与母鸡情敌的搏斗，这么做是为了觅得良配——你的公鸡伴侣。最后蛋蛋工厂将会根据你的形容词选择，利用 ai 画出一颗特别地蛋，也就是你的孩子。在异世界中，酷似十字路口的地图也许会让你想到什么，而你需要在这里做出选择，或是通过答题获得线索，或是以受伤为代价以身试险，以此找寻真正的方向。

为了让玩家有更好的游戏体验，我们实现了鼠标浮窗、hud 指令、游戏内音量调节、画面 4：3 与 16：9 的更换选择、本地存储，以及一些动画的制作，尽可能地接近真实地游戏体验。

《捡蛋》中有充足的剧情说明以及教程指引，但如若你仍有疑问，可以随时跟连接 LLA 语言模型的“系统”提出疑问，相信我，你会得到满意的解答的。

2 项目实施

2.1 场景

2.1.1 DynamicWorld 主世界

游戏的主世界，小鸡在此完成大部分的任务。这里着重介绍地图的随机生成：

创建地图前，玩家可选择决定地图生成的随机数种子，然后使用 python 的 random 库生成随机数实现地图随机生成。玩家恒定出生于(0, 0)，随机生成两个方向向量和长度，生成两段草径，并在第一个方向向量的末端生成一个狐狸塔、第二个方向向量末端固定生成一个大鸡圈。地图中的狐狸、米粒、母鸡等均在游戏开始时由随机数决定位置，游戏过程中由随机数决定随机游走方向。

(a)小鸡的家　　(b) 母鸡鸡圈　　(c)狐狸塔

图 1　DynamicWorld 地图元素展示

(a) 方向 1　　(b) 方向 2　　(c) 方向 3

图 2　DynamicWorld 随机种子地图生成展示

2.1.2 WitchWorld 异世界

老巫婆的世界中善恶并存，唯有得到长老教化的小鸡方能诞下通灵性的小鸡。老巫婆鸡可以帮助主角小鸡接受教育，但是前提是你要躲避狐狸的追踪和袭击，寻找线索并分辨出假长老鸡。

(a) 异世界地图　　(b) 答题页面

图 3 WitchWorld 部分展示

2.1.3 场景镜头移动

所有 World 类世界（即上述两场景）实现镜头跟随：镜头在游戏中以一个向量存储其位于游戏地图上的位置。渲染时，根据地图缩放比例，仅筛选出位于窗口可视范围内的实体和方块渲染，每个元素通过自身位置、镜头偏移和地图缩放比例，计算出在屏幕上的位置，随后渲染。

并且我们额外实现了，即在镜头跟随中建立了交互：

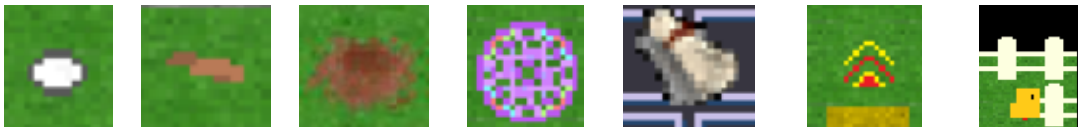
1）按下空格键，将镜头跟随的实体移到镜头中心，若已在中心则镜头跟随关闭。

2）按住鼠标中键滚轮可以拖动地图。

3）鼠标左键单击地图上的实体，令镜头跟随被单击的实体。

2.1.4 互动物品

互动物品在 World 存在，有多种互动物品，照片展示如下。



(a) 米粒　(b) 树枝　(c) 鸡窝　(d) 传送门　(e) 线索　(f) 神奇的墙　(g) 栅栏

图 4 交互物品

2.1.4.1 各交互物品介绍

(1) 米粒：Entity 类，当主角与米粒距离小于特定值后，米粒被自动捡拾，并且被从世界移除，米粒数量会转化为小鸡（玩家）的生命值。

(2) 树枝：Entity 类，与米粒几乎一样。树枝数量会转化为玩家背包中的树枝数量。

(3) 鸡窝：Entity 类，收集 100 树枝后，玩家可在家中键入 H 建造鸡窝。

(4) 传送门：Block 类，当玩家在该方块上停留满 3 秒，传送启动，玩家会被传送到异世界。

(5) 线索：Entity 类，类似米粒，但不会改变小鸡属性，只会调出一个 Window 窗口，并且被从世界移除。

(6) 神奇的墙：Block 类，叠在 Map 的 Block 上，允许玩家通过，Enemy 不可通过。

(7) 栅栏：Block 类，叠在 Map 的 Block 上，谁都不能通过，装饰的作用更大。

2.1.4.2 交互实现说明

实体被创造后，其 tick 函数会在每一帧被调用，该函数中有实体坐标和玩家坐标的距离计算，并检测是否小于特定值，若满足，则执行对应操作，如调用 player 中的 grow 函数来实现吃了一粒米后玩家生命值 1 点的增长。

2.2 角色

2.2.1 主要角色

小鸡！主要角色是由玩家控制的小鸡。

图 5 主角小鸡，代入玩家

2.2.1.1 移动控制实现原理

玩家通过 wasd 四个按键可以操控小鸡进行移动，其中四个方向的走动配有不同的图片。小鸡拥有自己的坐标，因为最后根据坐标被渲染到 Map 上，所以我们按键的控制本质是对小鸡 x、y 坐标做增量加减，每一帧检测按键是否按下，按下则执行加减。此外，同时按下 Ctrl 和方向我们使加减的增量翻倍，同时按下 Shift 和方向，我们使加减的增量减半。

2.2.1.2 小鸡自有属性

小鸡有自己的生命值、背包树枝值、血条、成长值、技能包。展示在了每个 World 的左上角以及状态面板里，状态面板可用 E 键调出。小鸡成长值首次到达最大值 100，视为小鸡长大了。

图 6 这张图既可以看到血条又能展示面板

2.2.2 友好 NPC

2.2.2.1 老巫婆鸡

存在于异世界，我们让老巫婆鸡任意的移动、但极为缓慢，等待玩家答题得线索后找到它，确认并找到它后，它会作为传送门送小鸡回到主世界。这里主要通过坐标和 random 模块实现任意方向的移动，然后还是 tick 函数中的距离计算，检测玩家并传送。

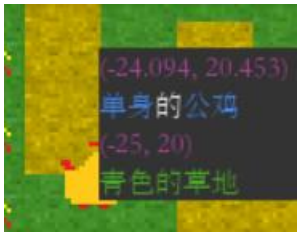
为了让阅读此报告的人也能体验辨别真假长老鸡，就不展示长老鸡地图片了。

2.2.2.2 Strong 的公鸡

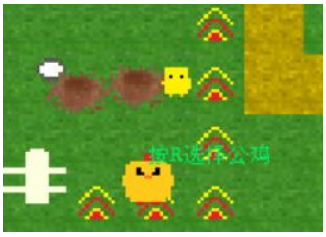
朋友们，你还见过比图片上的公鸡母鸡更加恩爱的夫妻吗？他们有自己的一个鸡窝，只会在自己的鸡窝周围一定范围内运动。公鸡不具有血量和战斗力，Peace and Love。

图 7 恩爱的鸡夫妇和他们的鸡窝

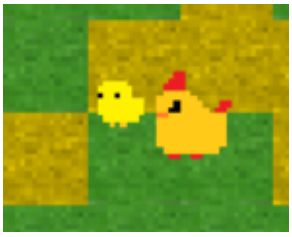
但小鸡也该拥有自己的爱情，所以玩家的任务是杀死一只母鸡，然后带着单身的公鸡回家，下蛋。公鸡对玩家的跟随与下文敌人对玩家的追踪实现方法除速度计算以外一致，下文会介绍。



(a) 失去母鸡后变成单身狗



(b) 所有丧偶的公鸡都可被选择



(c) 公鸡跟着小鸡乖乖回家

图 8 如何将一只公鸡带回家

2.2.3 敌人

2.2.3.1 蠢蠢的狐狸

蠢蠢狐狸是出现频率最高的敌人，他们守卫在主世界的狐狸塔，也在老巫婆世界出现。一样，还是 tick 函数检测玩家，一旦距离小于 A，狐狸开始以玩家实时所在位置为目标位置开始移动；一旦距离小于 B，调用 Player.damage( )函数，实现对小鸡血量的扣除。

图 9 蠢蠢的狐狸在盯着你看

2.2.3.2 丰腴的情敌

情敌母鸡也可以攻击玩家，且战斗力更强，攻击的能力同狐狸一样实现，仅设置了不同的参数，不过多赘述。

根据剧情，我们只有把母鸡杀了，小鸡才能上位。因此，进攻系统出现了。详细的进攻实现我们会放在创意当中介绍，这里介绍 Enemy 部分的被进攻相关部分：每只母鸡都有自己独立的 hp 血量，你可以通过将鼠标放在该鸡上看到血量剩余。血量的创建和加加减减写在 Damageable 类（继承 Entity 类）中，Player 继承故能被伤害，Enemy 类也继承，也支持被伤害。

小鸡长大前（即成长值首次到达 100 前），除非小鸡攻击母鸡，否则母鸡不会主动攻击小鸡。小鸡长大后，一旦玩家步入母鸡 2 曼哈顿距离以内，母鸡就会以飞快的速度追向小鸡并发起攻击。玩家离开范围后，母鸡会回到自己的窝附近。

图 10 丰腴的母鸡

2.2.3.3 假长老鸡

剧情需要，假长老鸡作为“错误选项”的身份在异世界存在。他们的伤害不同于狐狸与母鸡，只会对玩家造成一次伤害，用 bool 类型的 flag 实现。

(a) 长老鸡 1 号　(b) 长老鸡 2 号　(c) 长老鸡 3 号　(d) 长老鸡 4 号

图 11 老巫婆鸡正藏匿于四只长老鸡中

2.3 游戏机制

2.3.1 核心机制

玩家是一只被原主人抛弃的小鸡，它唯一的愿望就是长大并生下一颗厉害的蛋，也就是本游戏的终极目标——产蛋。

我们为终极下蛋任务做了铺垫和细化，变成了五个任务：吃米长大——捡枝造窝——得长老教化——觅得良配——许下对蛋的心愿。任务会逐一解锁，也需逐一完成，解锁全部任务，通关游戏并得到一颗独一无二的蛋 的图片，供玩家保存。

在世界窗口下，玩家键入 Tab 可调出任务面板。任务在未解锁的时候不能查看，上一项任务未完成，下一项任务就不会解锁。这里的任务页面切换，是由页面类的变量，等于不同数时渲染不同文字，来实现的——好用的 if。

图 12 任务面板

2.3.2 碰撞系统

原设计案中，只检测运动起始点和终止点的碰撞是完全不严谨的，速度较快时极有可能穿墙而过；但检测方块路径移动碰撞会使代码过于复杂。为简化模型，我们将所有实体均视为点，多个实体之间的临近判定改为实体位置的曼哈顿距离判定；实体与方块的碰撞采用较为严谨的判定系统。

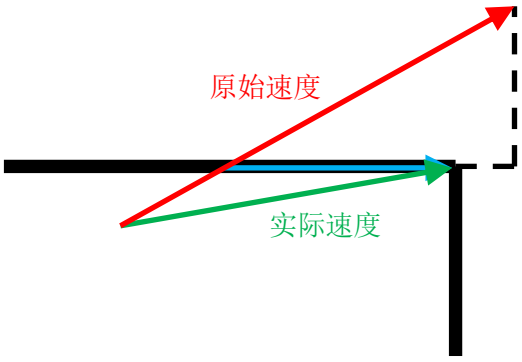


图 13 原理示意图

在 World 类中有一个 rayTraceBlock 函数，可以一次性获取由起始点开始、特定方向、特定长度内，所有经过的方块和与方块的碰撞点。遍历其返回列表，即可获得实体移动时碰撞到的方块。特别地，当实体撞到方块边界或方块角落时，如果有剩余的速度，游戏会将剩

余速度投影到撞击后实体仍然能够移动的方向。

2.3.3 资源系统

游戏的资源管理交由 Texture 类和 ResourceManager 类实施。ResourceManager 存储了所有注册的 Texture 实例，避免了频繁的文件读写，每个 Texture 实例对应唯一 ID 即资源文件路径，并分别选择性地适应了三种渲染方式（Map、UI、System 三种），针对不同需求，调用 Texture 类的对应渲染函数即可实现。需要自定义渲染时，也可以调用 getSurface 直接获取 pygame 表面进行相应的绘制工作。

2.4 LLM

2.4.1 对话系统

键入 Enter 即可跳出对话窗口。

图 14 对话系统窗口

此处我们让 LLA 语言系统作为一个“系统”存在，（感觉需要在最开始的 plot 最后强调一下 有任何难题可以像 LLA 询问）我们训练它说话简短，不回答与游戏无关的问题，并通过让它了解游戏的内容与玩法、以及所有键入的用法，来训练它成为一个系统。

在此界面中，单击输入框开始输入，输入框适应了中文输入法，可以使用左右方向键调控输入位置，可以使用 Backspace 和 Delete 键删除字符，长按可连续删除。

对话内容过多时，可以使用鼠标滚轮上下滚动，回顾过往的对话。

2.4.2 决策系统

Ai 生成 20 个形容词来描述蛋，与预设的约 10 个形容词一并随机选取其中一共 20 个，让玩家拖拽词语，选择其中的部分，然后 Ai 用选出来的词语，自己来画一个颗蛋（我们有基础的素材，ai 实现挑选素材、给素材选颜色）

限于语言 AI 的不完备性，AI 给出的回复有时不能很好地适应格式需求。此时我们会重新将记录发给 AI，尝试让 AI 重新生成一份合适的回复。语言 AI 对于图像、颜色等的处理稍逊色，但我们已经努力令其生成合适的结果，并将它选定的素材和颜色按照合适的次序，以随机数生成位置的方式组合，并呈现在玩家眼前。

(a) 选择页面　　(b) 蛋蛋生成页面

图 15 Ai 决策部分展示

2.5 游戏性

很明显，我们是真的想做一个体验感良好的游戏，合理的剧情、舒适的交互、清晰的规则，我们实现每一个功能都并非堆砌，是实打实地玩家更好的用户体验。

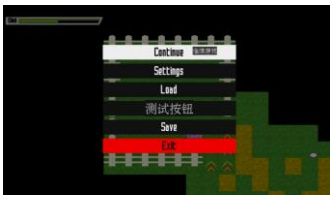
2.5.1 菜单

我们的游戏有三种菜单，分别为游戏开始时呈现的主菜单、设置菜单、暂停菜单和死亡菜单。由图片渲染、文字渲染和按钮组件组成。

所有的按钮初始化时，给定 4 个参数，确定按钮组件相对于窗口的大小，随后由 onResize 函数根据窗口大小计算出每个组件的实际大小。每个组件中有位置、宽高、定位方式、名称、浮窗描述、是否可用、颜色、默认文字颜色等属性，配备 onMouseDown、onMouseUp、onHover、onTick 四个回调函数。每当鼠标传入操作时，系统判断鼠标的位置是否位于这些组件之上，然后根据鼠标操作调用组件相应的回调函数，实现事件响应。

(a) 主菜单

(b) 设置菜单



(c) 暂停菜单

(d) 死亡页面菜单

图 16 菜单合集

2.5.2 BGM

主菜单有一个音乐，每个世界有一个音乐，捡米粒织窝等事件都有各自的音效。

为了保证音乐顺利播放，我们通过统一的音乐管理器实现。背景音乐和音效被分开管理，因为考虑到我们对两者有明确不同的需求：一个一直播放，游戏开始到游戏结束之间始终播放，可能会随世界变化而变化，但不停；另一个随玩家的操作播放或暂停。

但实现的方式大同小异。各有一个表示开关的 bool 类、表示音量大小的 float 类（我们的音量可在 setting 里拖拽调整）、存有可能会播放的所有音乐的列表，以及播放状态列表。如我们为了实现暂停存档，播放状态列表会存放暂定的时间，没有暂停过就是 0。音效，因为 pygame.mixer.Sound 用法与 music 不同，music 一个时间只能有一个声音在播，sound 可多个同播，则 sound 状态列表放置在播的音效，以便于对任意一个精准执行暂停的命令。

2.6 代码

由于我们的代码基底搭建时间较早，因而没有完全采用提供的模板。参考模板，我们最终对基底做了较大的扩展和修改。

1、 游戏很大程度上仿照了 Minecraft 的代码结构和术语名称，例如 Block 方块即 Tile；Entity 实体即所有不可动实体、NPC 和 Player，World 世界即 Scene。
2、 采用多线程，分事件、渲染、游戏、异步四线程，分别处理玩家输入消息、渲染、游戏更新、AI 交互，提升游戏性能。
3、 舍弃了原有的 EventListener 机制，改为 tick 机制、信息取用机制（而不是直接响应），实现次序更严谨事件处理逻辑；将交互信息存储在 Interact 中，在 Window、World 等不同位置均可以获取所有的玩家交互信息，让玩家拥有更丰富的操作，和更简便的代码实现。
4、 不需要 Collidable 类；所有元素均可在 tick 中实现其自身的碰撞逻辑，主要是 MoveableEntity。
5、 所有 Box 类改为 Window 类，并定义了更丰富的组件和功能。
6、 Portal 类，改为世界上的一个“传送法阵”方块。
7、 和 Java 源文件结构，减少单文件的代码量和类的数量，并按不同功能将代码分装在各个文件和各个文件夹中。
8、 多态思维、更深度地面向对象，扩展了各个游戏元素的表现和行为丰富度。

另，各文件的具体说明已写在 readme.md，文末也附明了。

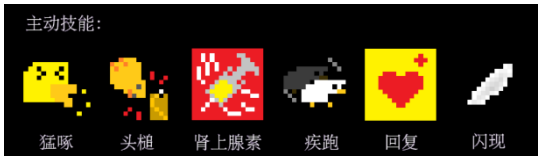
2.7 创意

2.7.1 剧情窗口

为了增加玩家沉浸式的体验，游戏设置了剧情窗口，通过游戏开始前的剧情介绍，让玩家沉浸式进入游戏的世界。在剧情窗口中，玩家可以通过按钮与窗口互动，逐页翻动剧情窗口或直接跳过。

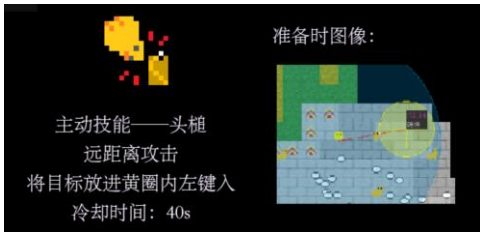
2.7.2 实时战斗系统

为了增强游戏性和趣味性，我们采用了实时战斗模式。玩家独享一个主动技能列表、一个被动技能列表，分别存储玩家拥有的主动/被动技能。由于技能机制的特殊性，此模块采用了事件响应的机制，在玩家受到伤害前后、tick 执行前后、成长前后、死亡前添加了事件队列。技能的获得是永久的，获得技能后，技能会在玩家的相应事件队列中永久地添加响应器来实现技能功能。例如被动技能屹立不倒，就是在 preDeath 死亡前队列添加响应器取消死亡，然后为玩家回复一定的生命值。主动技能都拥有 onUse，由 Player 类的 tick 函数检测玩家按键盘按键的行为，调用选中主动技能的 onUse 触发主动技能。



(a) 主动技

(b) 被动技



(c) 准备模式样式

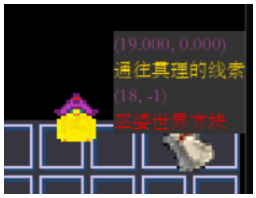
(d) 技能升级

图 17 技能相关展示

2.7.3 鼠标浮窗

在所有菜单界面，将鼠标悬浮在按钮上方时，按钮底色及文字颜色会进行对应更改，增强菜单的交互性。

(a) 主世界里



(b) 异世界里

(c) 菜单面板上

图 18 随处可见的鼠标浮窗

2.7.4 动画

利用游戏每一帧都会渲染，让相同的实体在不同帧按顺序渲染不同的图片，从而实现了动画感。无法在文档中展示，玩家可以在游戏中的织窝和教化时看到。

2.7.5 个性化设置窗口大小

考虑到 pygame 的渲染分辨率问题和玩家的电脑屏幕大小不同，初始窗口的大小一定是玩家电脑屏幕长款的各一半；随后允许玩家手动更改窗口的大小，以获取更佳的游戏视觉体验。我们对游戏做了 16:9 和 4:3 两种比例的适应，玩家可以在设置中调节选用的屏幕比例。

为了实现不同比例的渲染，游戏采用渲染器 Renderer 对 pygame 的渲染再封装，给每个 Texture 一个固定的缩放比例，给 Renderer 一个适应窗口大小的可变缩放比例，并将所有的渲染任务全部交由 Renderer 完成，窗口大小改变时统一重算表面缩放并保存，避免了每帧计算缩放导致的严重性能问题。

2.7.6 本地保存功能

考虑新手玩家的游戏流程不熟悉且前期难度较大，或是玩家希望保存当前生成的世界，游戏配备了保存的功能。使用 python 的 json 库，将 World 中所有的 Block，Entity（包括 NPC，Player 以及 Player 的技能）打包为字典保存在 json 文件中，玩家在存档加载界面就可以看到玩家先前保存的存档，存档过多时还可以上下滚动鼠标滚轮翻看。

2.7.7 随机数种子指定、世界名称指定

为了适应存档功能，游戏允许玩家在创建世界时指定游戏随机数种子和存档名称（也即世界名称）。留空时，随机数种子采取 python 的 time 库 perf_counter_ns()作为游戏种子，名称默认为“种子+序列世界”。如果玩家指定的种子不合法，或世界名称已存在，游戏会给出提示并拒绝玩家创建世界。极小概率下，如果默认名称重复，则在名称后添加“（1）”“（2）”确保一定不存在重复名称。

附件：

- assets/ 所有静态图像、声音资源、字体

- block/ 方块相关逻辑

- block.py 所有的方块类和子类

- manager.py 方块资源管理器，将方块类和方块 ID 一一对应，减少循环 import、局部 import

- entity/ 实体相关逻辑

- active_skill.py 所有主动技能类

- enemy.py 所有敌对生物类

- entity.py 实体基类，及玩家、蛋等末端实体类

- manager.py 实体资源管理器，将实体类和实体 ID 一一对应，减少循环 import、局部 import

- skill.py 技能基类及所有被动技能类-

- interact/ 交互相关逻辑

- status.py 状态类，用于保存和简化处理玩家交互信息

- LLA/ AI 交互逻辑

- ai_decision.py 自动生成词语，由玩家选择后，给出生成蛋的内容和颜色参数

- chat_with_ai.py 音像资源管理器

- music/ 声音、音效

- music.py 音像资源管理器

- render/ 渲染逻辑

- font.py 管理所有字体资源

- renderable.py 渲染基类，及所有可渲染对象

- renderer.py 渲染器

- resource.py 管理纹理图片资源

- save/ 存档逻辑

- configs.py 处理游戏配置文件，保存玩家设置

- save.py 处理游戏存档数据

- user/ 玩家信息。由游戏自动生成，首次运行前不存在

- archive/ 所有存档文件

- config.json 游戏配置文件

- utils/ 所有工具模块和类工具模块

- element.py 游戏元素基类。与 Item 协作，现可弃用

- error.py 游戏内定义的错误类

- game.py 游戏框架逻辑，游戏管理器，相当于源码的 GameManager

- sync.py 用于防止多线程数据冲突造成游戏进行不协调

- text.py RenderableString 类，简化文本渲染流程

- vector.py 向量类，提高位置计算相关代码可读性和流程

- window/ 窗口相关逻辑

- hud.py HUD 类，在游戏中实时现实游戏信息，包括血条、成长值、文字提示等

- ingame.py 游戏内窗口，包括状态窗口、任务窗口等

- input.py 输入窗口，主要包含 AI 助手窗口

- widget.py 窗口按钮

- window.py 窗口基类，及开始窗口等游戏流程外窗口；鼠标浮窗

- world/ 游戏世界（即场景）相关逻辑

- world.py 所有世界（场景）类

- *main.py* *游戏入口点*