**期末大作业\_2组**

**Microsoft VS Code**

作者：姜竣瀚 10255102512 李福洲 10255102515 梁家伟 10255102506 凌菲阳 10255102506

**摘要：**

本项目是一款以 “Microsoft VS Code” 为灵感来源的类植物大战僵尸游戏，名称巧妙呼应经典游戏《Plants vs. Zombies》。游戏核心玩法为玩家在 5×10 网格中布置 16 种 “IDE / 编程工具” 类植物，消耗 “学分” 资源，抵御 10 种不同类型的 “BUG” 僵尸分波进攻。游戏包含两章主线关卡及终章剧情过场，通过线性解锁机制逐步提升难度，玩家需合理规划防御布局与资源分配，守护左侧安全线不被僵尸突破，最终完成通关。

**一、游戏的核心介绍** ：

（一）主题与剧情

游戏以 “代码编写过程中 BUG 入侵编辑器” 为核心主题，构建沉浸式剧情线：BUG 化身僵尸突破编辑器防线，玩家需操控各类 IDE 工具植物展开抵御，剧情分为两章主线内容及终章过场，通过关卡推进逐步解锁剧情脉络。

（二）玩法循环

核心场景：5×10 网格防御区域，僵尸从右侧分路向左侧安全线推进

植物系统：16 种不同类型的 IDE / 编程工具植物，涵盖输出型、抛射型、墙体型等，每种植物具备独立血量、伤害、射速、冷却时间及部署费用

资源机制：以 “学分” 作为核心经济单位，部分植物可生成学分，植物部署需消耗对应学分且受冷却时间限制

战斗流程：僵尸分波次进攻，每关包含 3 波敌人，波次间隔与批次配置随难度动态调整，植物自动攻击范围内敌人，玩家需实时调整防御布局

（三）单位设定

敌人设定：共 10 种僵尸，基于 “BUG” 特性设计，每种僵尸在生命值、移动速度、击杀奖励学分等属性上存在差异

植物设定：按功能划分为输出类、抛射类、墙体类、经济型等，具备独立属性参数，部分植物无攻击能力但可提供防御或经济产出

（四）关卡与难度

关卡结构：两大主题章节，共 10 个关卡，采用线性解锁模式，未解锁关卡以灰色遮罩标识

难度计算：通过compute\_level\_difficulty函数基于关卡元数据（theme/d）计算难度标量，范围限制在 0.6-2.0

初始资源：初始学分通过starting\_money\_for\_level函数计算，与难度成反比，取值范围为 2000-3800

波次配置：每关包含 3 波敌人，波次间隔、敌人批次数量随关卡难度动态调整

（五）胜负条件

胜利条件：成功抵御所有波次敌人进攻，且场上无残留敌人，触发胜利动画与剧情播放

失败条件：任一僵尸突破防御抵达左侧安全线，触发失败音效与失败界面

（六）系统与界面

状态机架构：包含主菜单、关卡选择、植物选择、游戏内、暂停、胜利、失败、档案（图鉴 / 提示）、剧情、制作人员 10 种界面状态

存档机制：通过save.json文件记录关卡解锁进度，支持进度续存

资源管理：资源管理器动态加载图片、音效资源，提供占位符处理机制

运行参数：分辨率 1600×900，帧率 60fps

**二、主要函数（均在new\_game.py）：**

- `Plant.update`/`fire`：按行检测敌人并发射不同类型子弹（直线/抛物）。

- `Enemy.update`：推进、咬植物、死亡判定。

- `WaveManager.update`：按波次与间隔生成敌人队列，控制结束条件。

- `compute\_level\_difficulty` 与 `starting\_money\_for\_level`：根据关卡元数据计算难度和初始金钱。

- `Game.start\_game`：按选关与选卡初始化金钱、冷却、波次管理。

- `Game.update\_gaming`：一局核心循环，刷怪、碰撞伤害、胜负判定、放置/铲除、UI 绘制。

- `Game.run` 及各状态更新（`update\_main\_menu`/`update\_level\_select`/`update\_plant\_select`/`update\_pause`/`update\_win`/`update\_lose`/`update\_story`/`update\_archive`/`update\_credits`）：驱动主事件循环与界面状态机。

**算法、美工及ai应用：**

**算法：**

· 植物攻击算法（Plant.update：95-137 行）

每帧筛选同路敌人，获取距离最近目标

经济型植物定时生成学分，墙体类及特殊植物无攻击功能

攻击型植物按射速触发fire函数生成子弹，支持直射或抛物两种攻击模式

· 子弹生成算法（Plant.fire：139-177 行）

计算子弹初始坐标，抛射型子弹下调水平 / 垂直初速度

针对目标敌人采用简单物理模型估算抛物落点

实例化Bullet对象完成子弹生成与发射

· 僵尸行为算法（Enemy.update：205-247 行）

检测与植物的碰撞关系，同路重叠时停止移动并啃食植物（按帧扣除植物血量）

植物被摧毁后僵尸继续前进，否则按设定速度向左移动

包含僵尸死亡判定与状态切换逻辑

· 波次管理算法（WaveManager.update：355-430 行）

依据波次配置（总数、批次、间隔、波间歇）决定敌人生成时机

按批次随机生成 1~batch 数量的敌人，随机分配行进路线，按波号匹配对应敌人池

单波结束后进入间歇期，所有波次完成后设置finished\_spawning标识

· 难度与资源计算算法（460-480 行）

通过关卡主题（theme）与难度参数（d）调整难度标量，限制在 0.6-2.0 区间

初始学分随难度反比缩放，通过数值夹紧确保在 2000-3800 范围内

· 游戏核心循环算法（Game.update\_gaming：1027-1178 行）

帧循环流程：刷怪倒计时→更新植物 / 子弹 / 敌人状态→碰撞检测与伤害计算→学分奖励发放

胜负判定：实时检测波次是否全部结束且场上无敌人（胜利）、僵尸突破安全线（失败）

UI 交互处理：植物放置 / 铲除、冷却遮罩显示、金钱不足提示、鼠标跟随图标更新

· 状态机驱动算法（Game.run+ 状态更新：650-930 行）

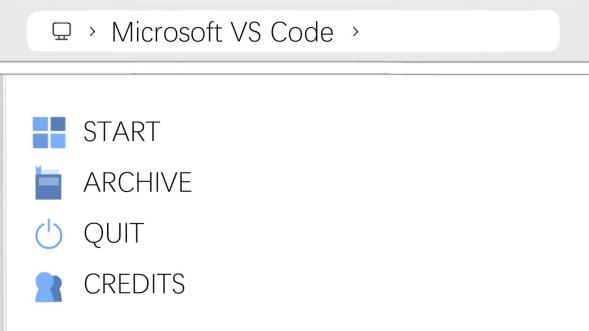
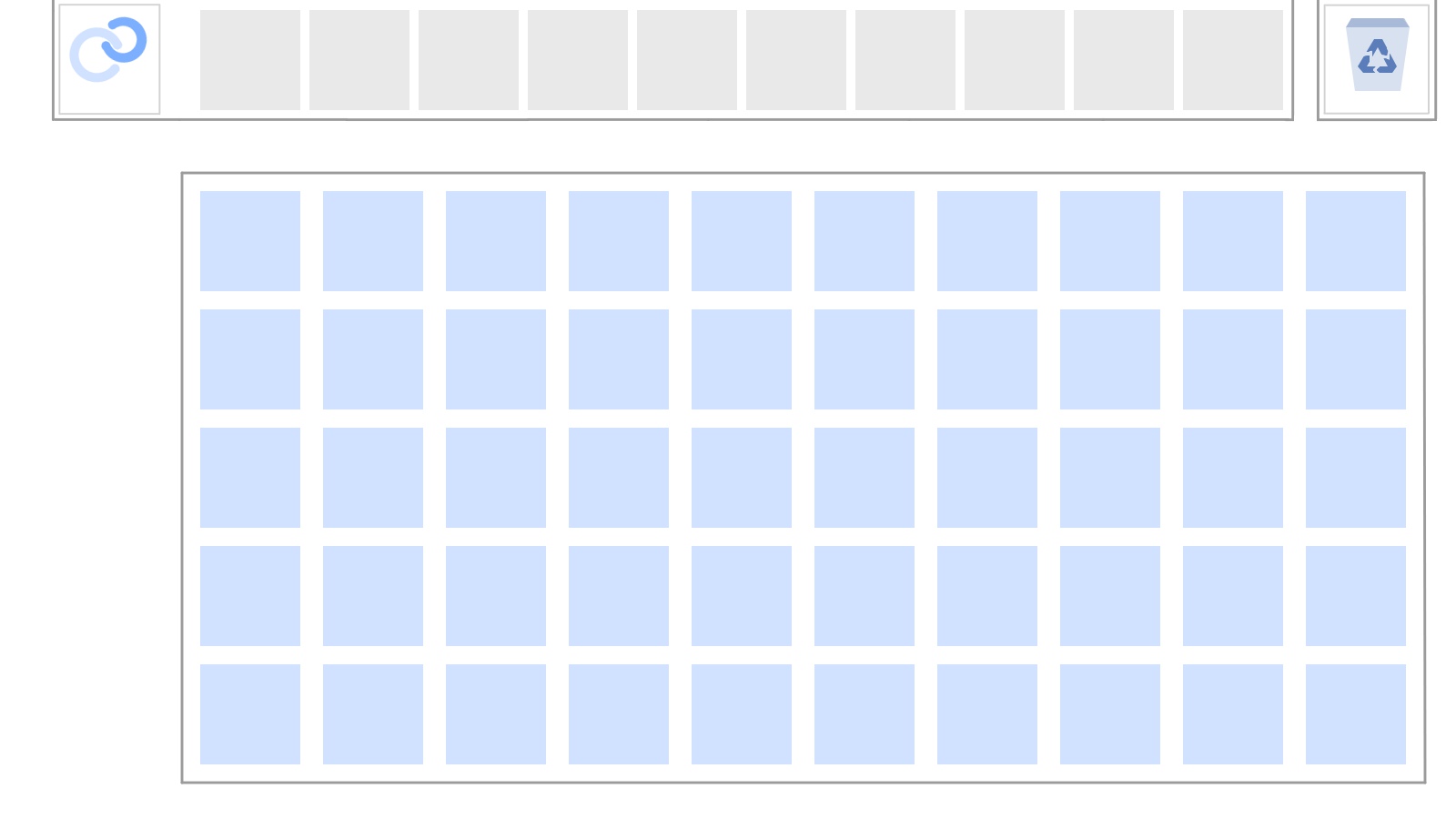
主事件循环按当前状态分发至对应界面处理逻辑（菜单 / 选关 / 剧情等）

状态切换后短暂屏蔽鼠标输入，避免误操作

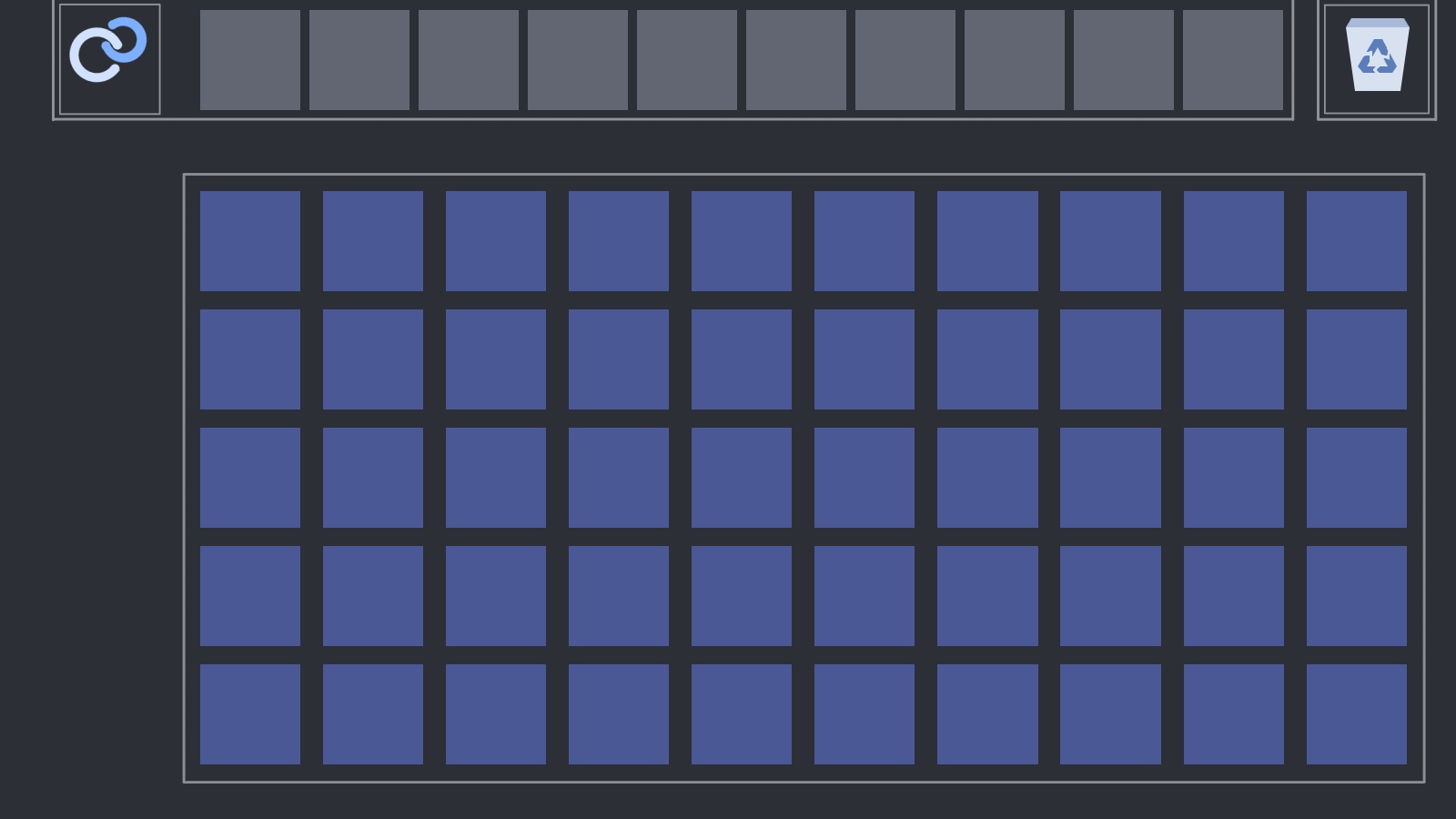
支持跨状态交互（如游戏内暂停、剧情跳转、图鉴查阅）

**美工：**

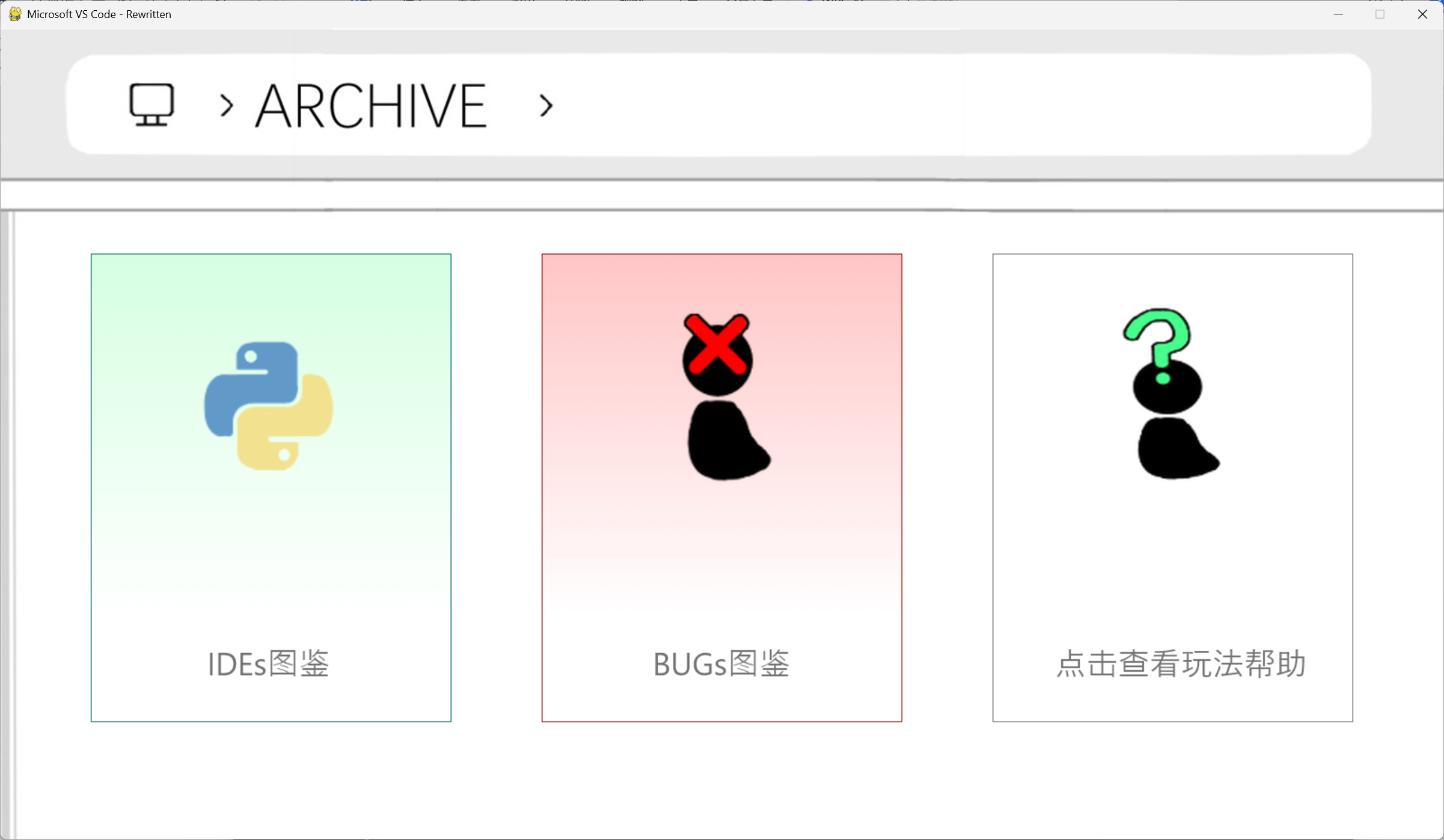
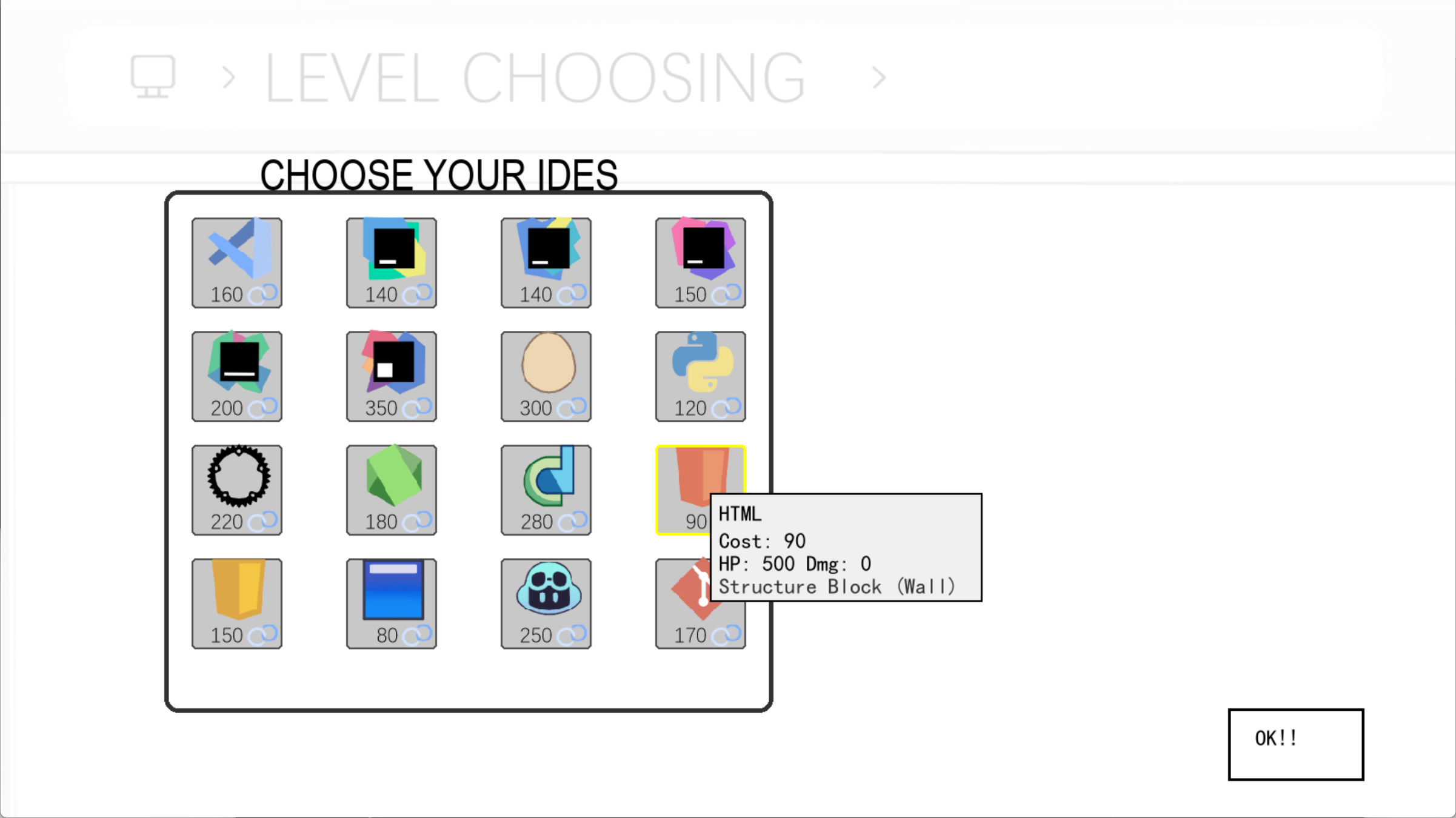
UI灵感来源于win电脑桌面以及编辑器的黑色背景。纯手画无ai（给美工跪了）。主要UI如下（所有美工部分均能在/resource中找到）

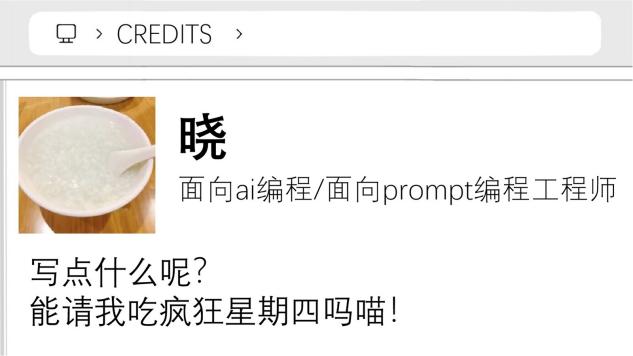
主页面 第一章地图

第二章地图 选关界面（未解锁的关卡灰色遮罩）



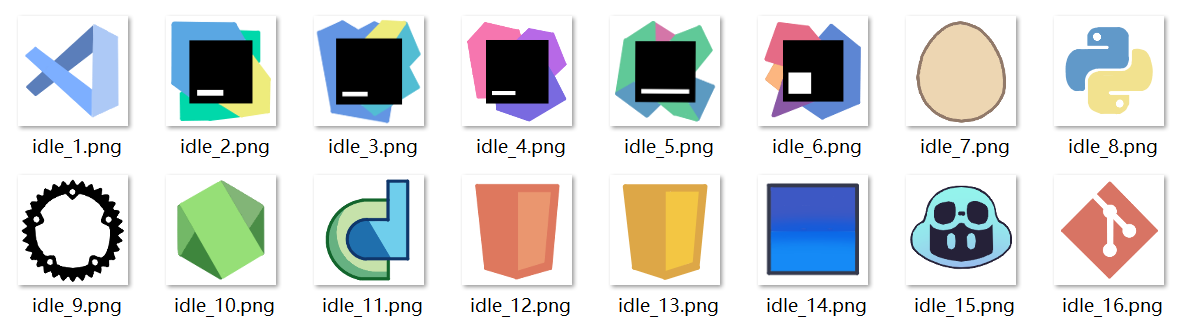
选择植物页（鼠标移动到ide上方时悬浮显示介绍） 图鉴/教程页



CREDITS介绍开发者页 游戏画面

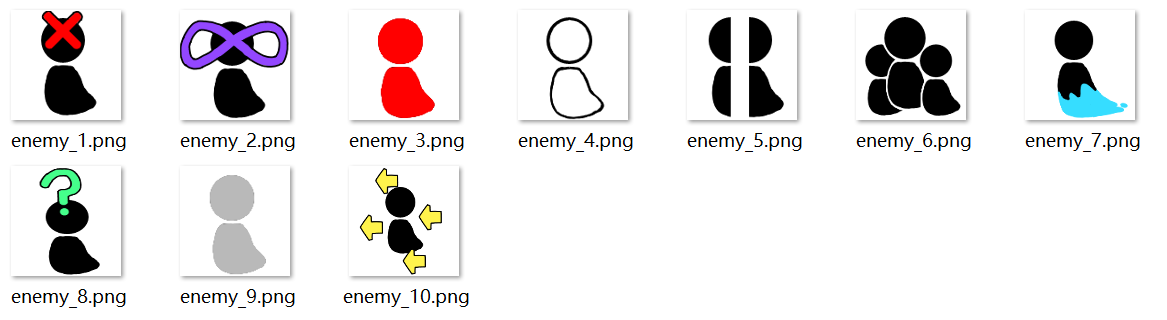
IDEs（植物）BUGs（僵尸）：

植物（IDEs）：16 种植物均以编程工具 / IDE 为原型设计，对应不同功能定位。



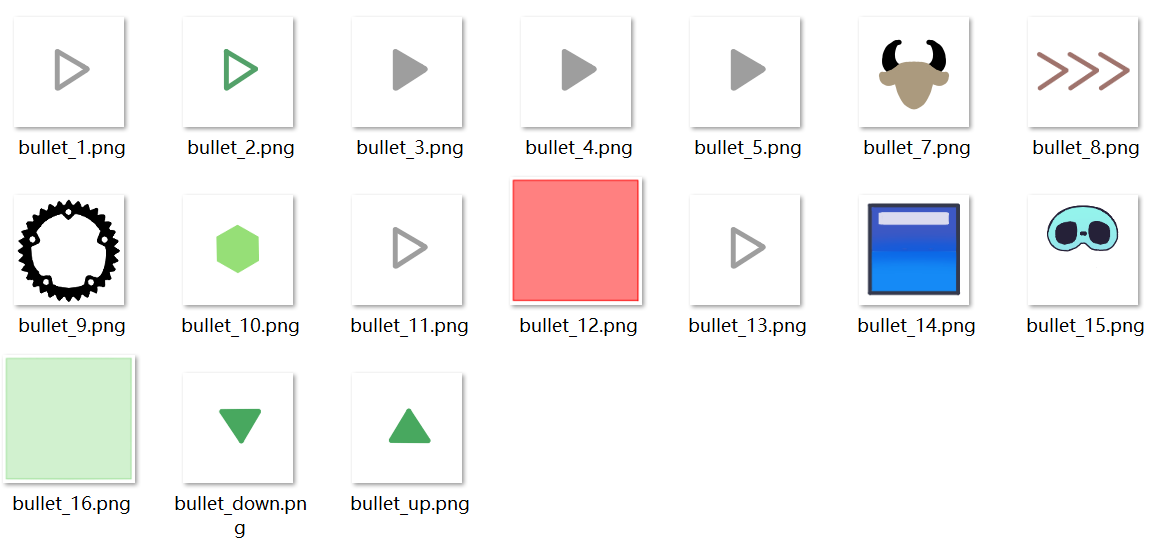
Ides

僵尸（BUGs）：10 种僵尸基于不同类型的编程 BUG（语法错误、内存溢出等）设计，具备差异化视觉特征



Bugs

子弹灵感来自于coding用到的各种符号/ide自带的按钮



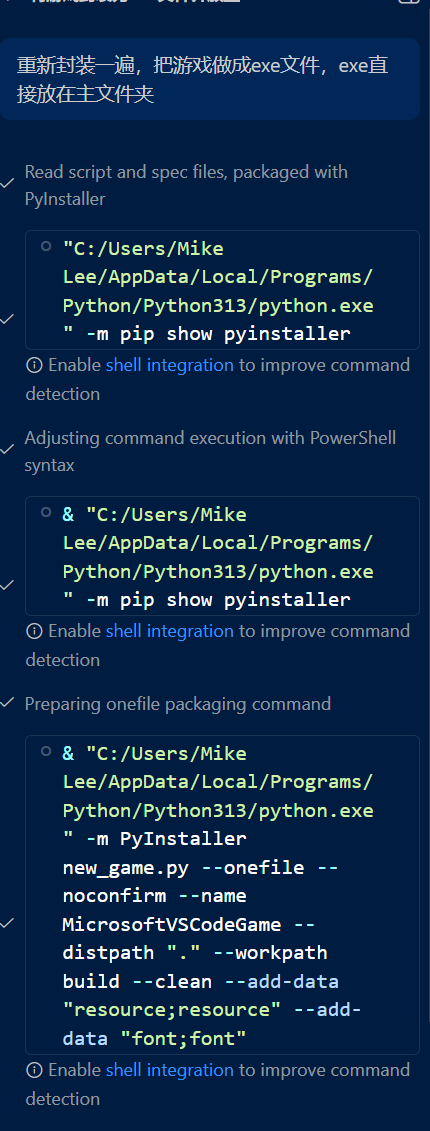
子弹

**AI应用：**

毫无疑问ai在我们的游戏制作中帮了大忙，由于各种原因纯手搓完成这个游戏基本上是不可能的，于是我们手写了游戏的基础模型（具体第一版代码详见Microsoft VS Code.py）极其粗糙，没有关卡系统没有选择植物，由于没有使用面向对象导致可读性也不好，实现了游戏基础玩法的运行。所以我们使用gemini3pro重构了我们的基础逻辑，使用面向对象以增加可读性、方便debug等。

在重构完基础逻辑之后，判定胜利/失败、选关、选择ide、图鉴/帮助/开发者页面以及音效控制都交给了GPT-5.1-codex-max。

以上均使用vscode内github的copilot系统实现，对开发而言非常的方便省力，甚至可以在人工审批之后直接执行命令（例如将游戏打包成exe文件就是让它自己完成的）



**心得体会：**

首先，AI 的发展速度给我留下了极其深刻的印象。与几年前对话能力尚显生硬、理解能力有限相比，如今的 AI 已经能够参与项目开发，完成相当一部分实际工作，并给出整体上令人基本满意的结果，这种技术迭代的速度令人惊叹。然而，AI 仍然存在明显不足，其“智能”程度并不稳定。例如，在代码重构过程中，我已经明确要求将主程序拆分为多个 Python 文件，但 Gemini 仍然忽略这一约束，将一千多行代码集中写入 new\_game.py 中，降低了代码的可维护性。此外，AI 在准确性方面也存在问题，Codex 在编写新功能时有时会产生不会直接报错、但会严重影响游戏逻辑的隐藏 Bug，因此仍然需要人工进行审查和修改。

另一方面，我逐渐形成了一种新的认识：在 AI 时代，真正具备竞争力的将是“善于使用和理解 AI 的程序员”，而不仅仅是会写代码的程序员。前者更接近于计算机科学家的角色，而后者在未来可能会被 AI 部分取代。不过，在实际的游戏开发过程中，我也清醒地认识到了自身的不足，尤其是在代码基础和工程能力方面仍有较大提升空间。与此同时，这次项目实践也显著提高了我使用 AI 的能力，使我对如何更有效地引导和“驯服”AI 形成了自己的理解。

1. **分工及心路历程**

李福洲：策划、ai代码构建、debug、报告撰写

姜竣瀚：基础逻辑编写，策划

梁家伟：策划、美工

凌菲阳：主美工、策划

为什么要做一个类似于pvz的游戏呢?摘要已经说明灵感来自于vscode跟植物大战僵尸的名字很像，但这其实只是一个原因。对我来说学习计算机是一趟充满挑战的旅途，它并不全是让人充满快乐的路程，记不住语法、看不见bug、搞不清逻辑等问题直到现在还困扰着我。我不能说自己是一个很优秀的学生，但我确实在努力。我们的游戏中ide打怪也代表着我对与学好计算机的期待吧，虽然它不完美，但它是我的第一个游戏项目但肯定不是最后一个。

下面是做这个游戏的大概历程：

初期：确定创意方向，明确分工与时间节点，低估项目复杂度导致进度滞后

中期：集中攻关核心功能，1 月 8 日 - 9 日夜间持续调试至凌晨，重点解决绘制异常等顽固性 BUG

后期：整合功能模块，优化游戏体验，完成打包测试与报告撰写

团队感悟：感谢团队成员的协作配合，尤其感谢美工团队的手工创作，同时感谢 AI 工具在代码重构与功能开发中的高效支持，使项目得以顺利完成

1. **不足以及改进**

玩法多样性：植物攻击方式单一，缺乏差异化策略选择

特效表现：第二章计划设计的特殊效果未实现

难度平衡：刷怪逻辑不合理导致难度曲线波动较大

资源管理：项目资源组织不够清晰，维护成本较高

界面美观：UI 设计简洁但视觉表现力不足，缺乏细节优化

改进：

功能优化：丰富植物攻击方式，增加 AOE、减速等特殊效果

难度重构：重写刷怪逻辑与难度计算算法，优化关卡难度曲线

资源整理：重新规划资源目录结构，建立规范化的资源管理体系

界面升级：优化 UI 视觉效果，增加动态过渡动画与交互反馈

项目沉淀：寒假期间完成完整项目重构，上传至 GitHub 开源平台，提升工程实践能力

参考文献暂无

**附录：游戏手册**

（一）基础控制

界面导航：按下 ESC 键返回上一层界面 / 游戏内暂停；暂停后按 ESC 返回主界面，按 ENTER 键回到游戏

翻页操作：在图鉴 / 帮助界面按方向键←→实现翻页

退出游戏：主界面点击 QUIT 按钮，或游戏内通过暂停界面返回主界面后退出

（二）植物部署与管理

部署操作：左键点击植物选择栏中的目标植物，移动光标至网格对应位置，再次左键完成部署；右键可取消当前选择

铲除操作：左键点击右上方垃圾桶图标，再点击已部署的植物即可铲除；右键可取消铲除状态

（三）界面说明

主界面：提供游戏启动、档案查阅、退出、制作人员查看入口

档案界面：包含 IDEs 图鉴、BUGs 图鉴、玩法帮助三个模块，可查看详细属性与操作指南

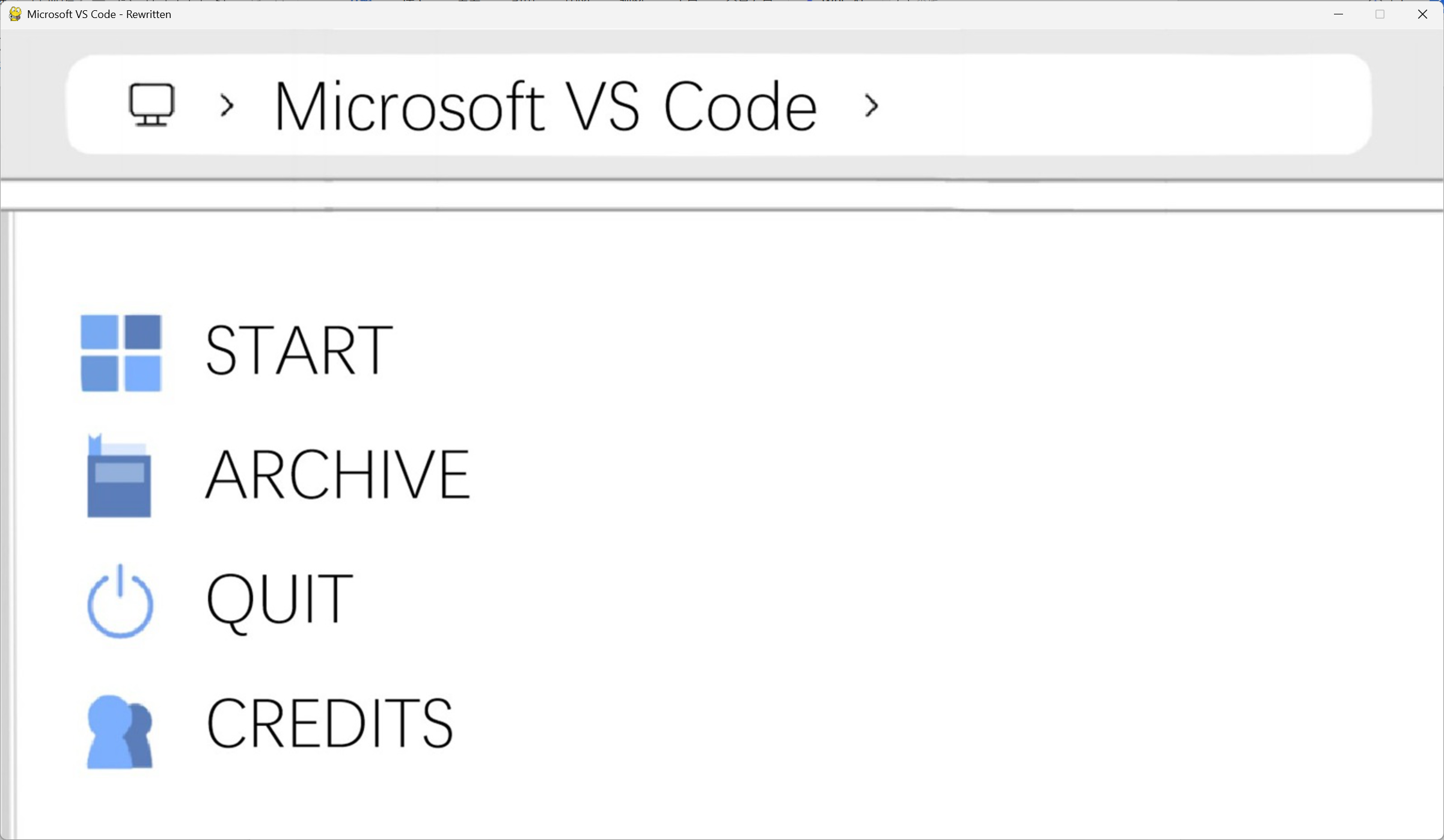
选关界面：展示已解锁关卡，点击对应关卡进入剧情与植物选择阶段

战斗界面：实时显示网格防御区域、植物状态、僵尸波次进度、剩余学分等核心信息

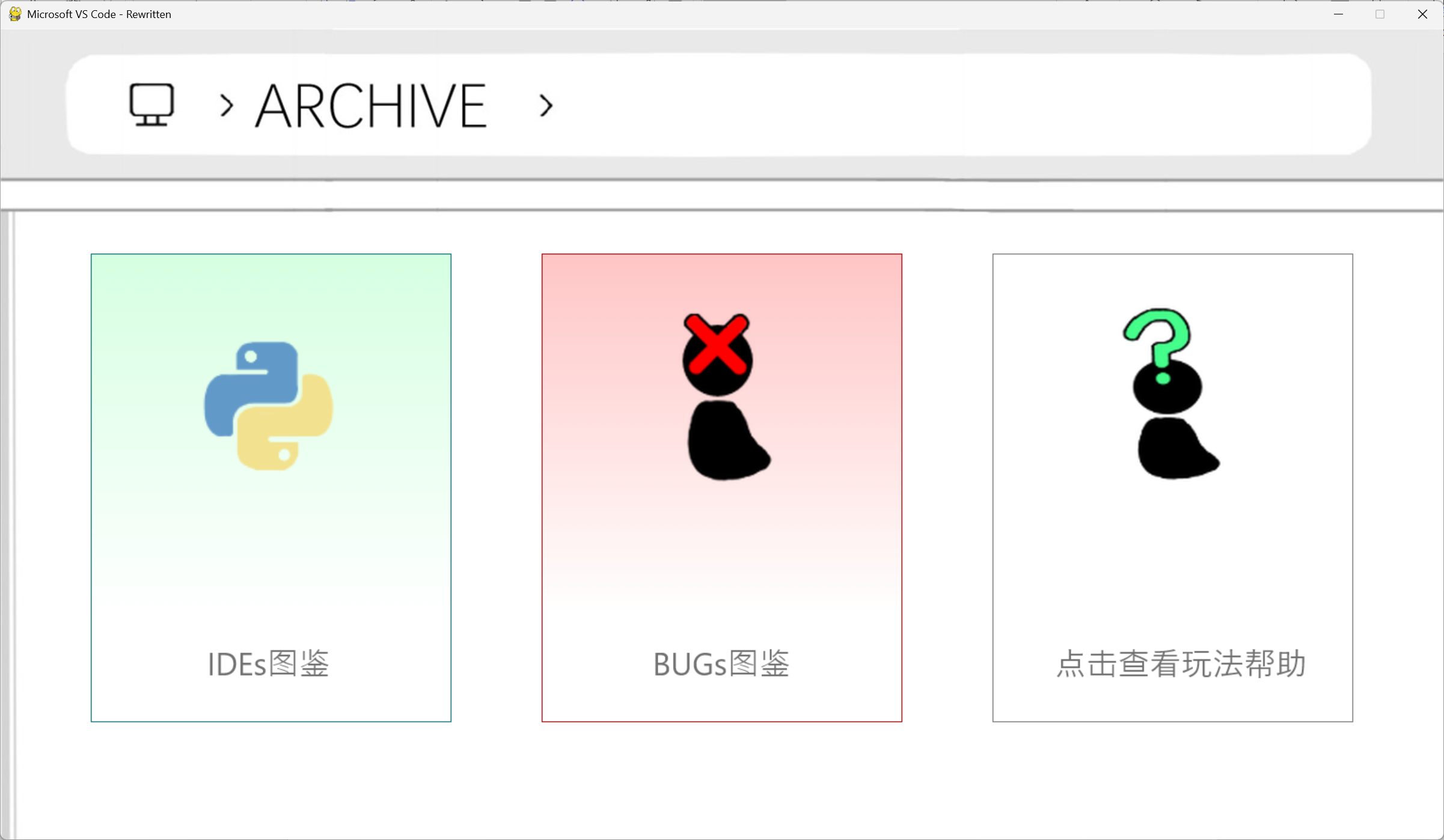
暂停界面：提供返回游戏、返回主界面选项，显示暂停倒计时提示

胜负界面：胜利时播放通关动画与剧情；失败时显示 ERROR 提示，按 ESC 返回主界面

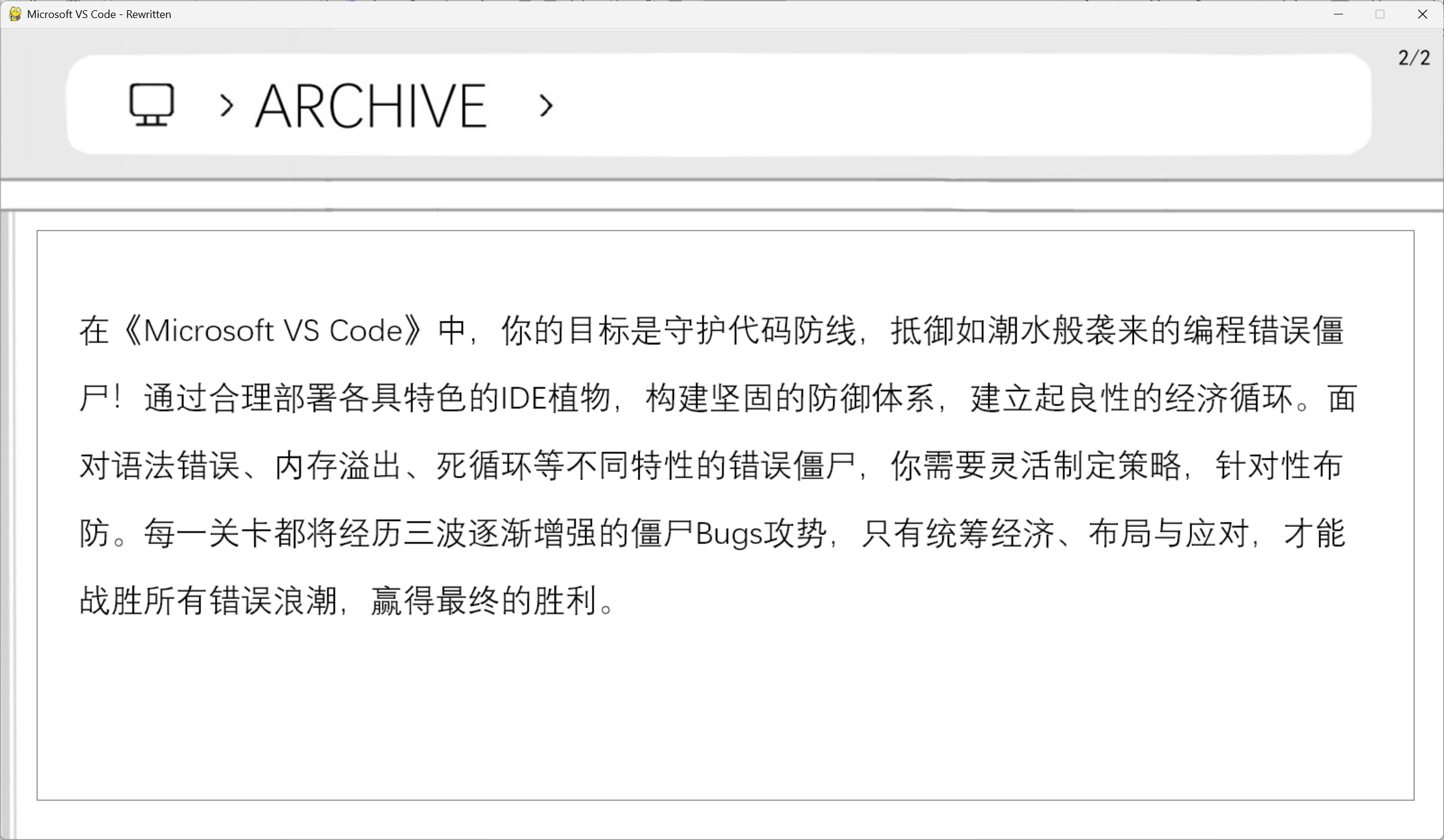
开始界面



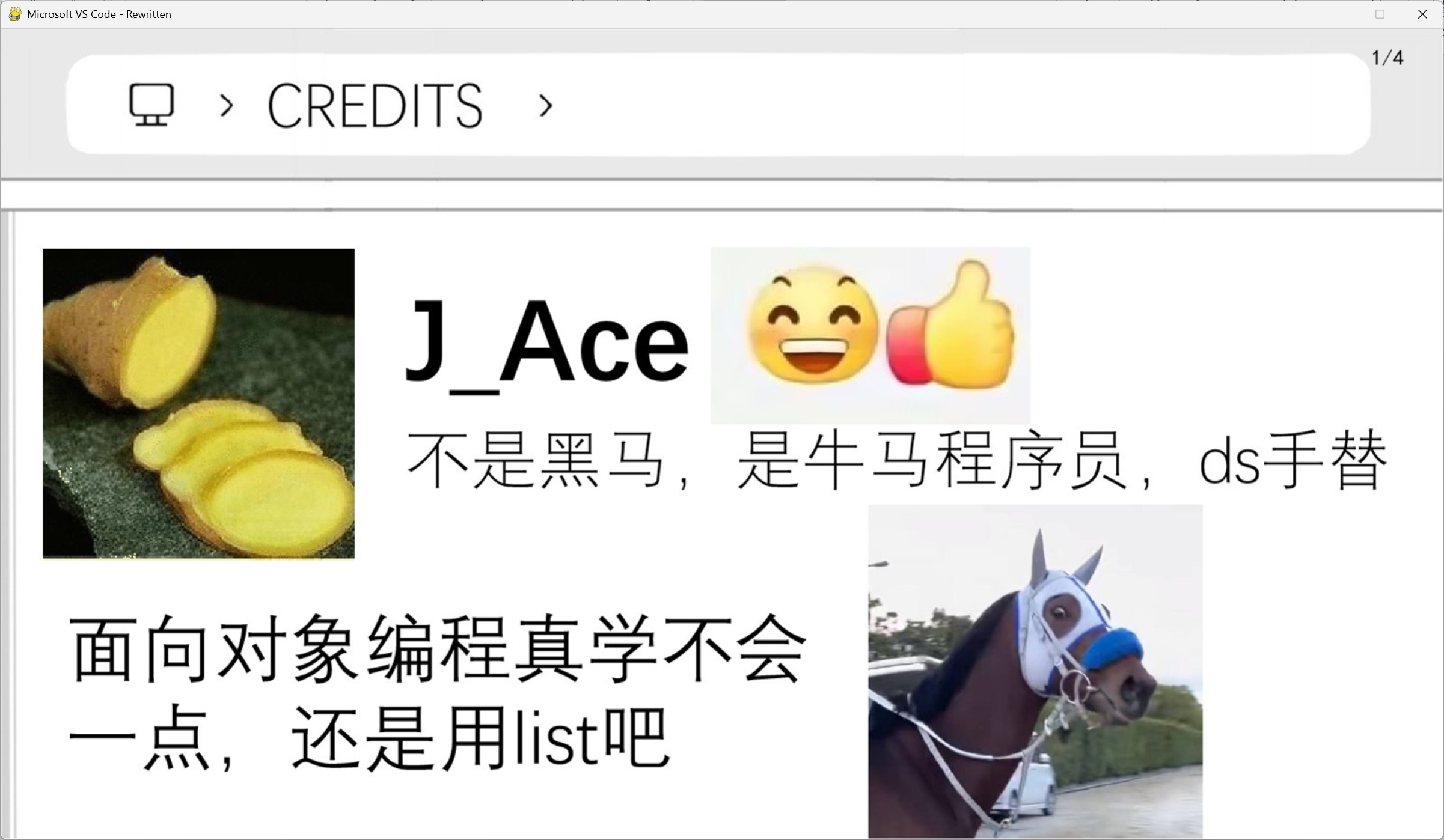
ARCHIVE

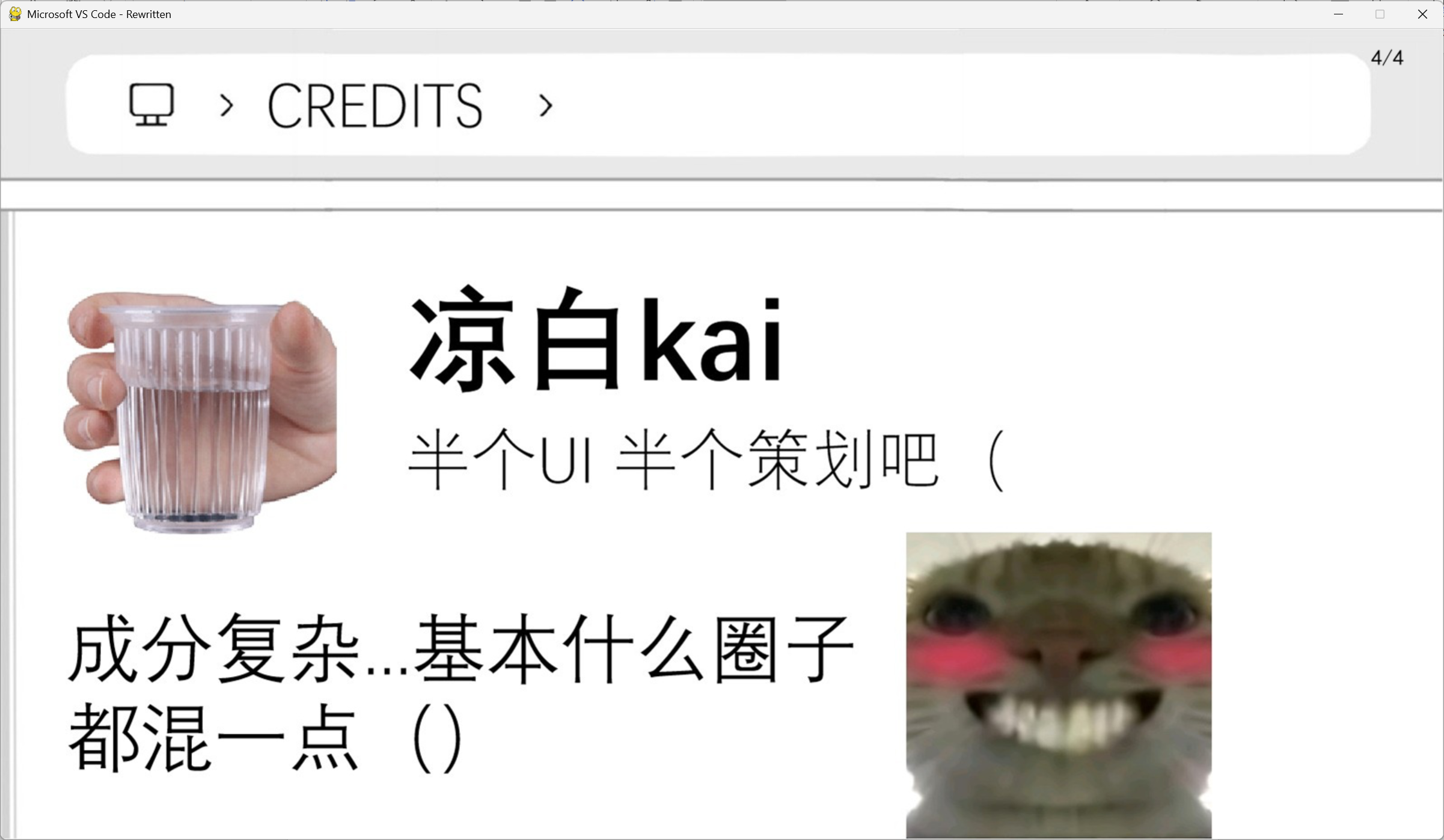
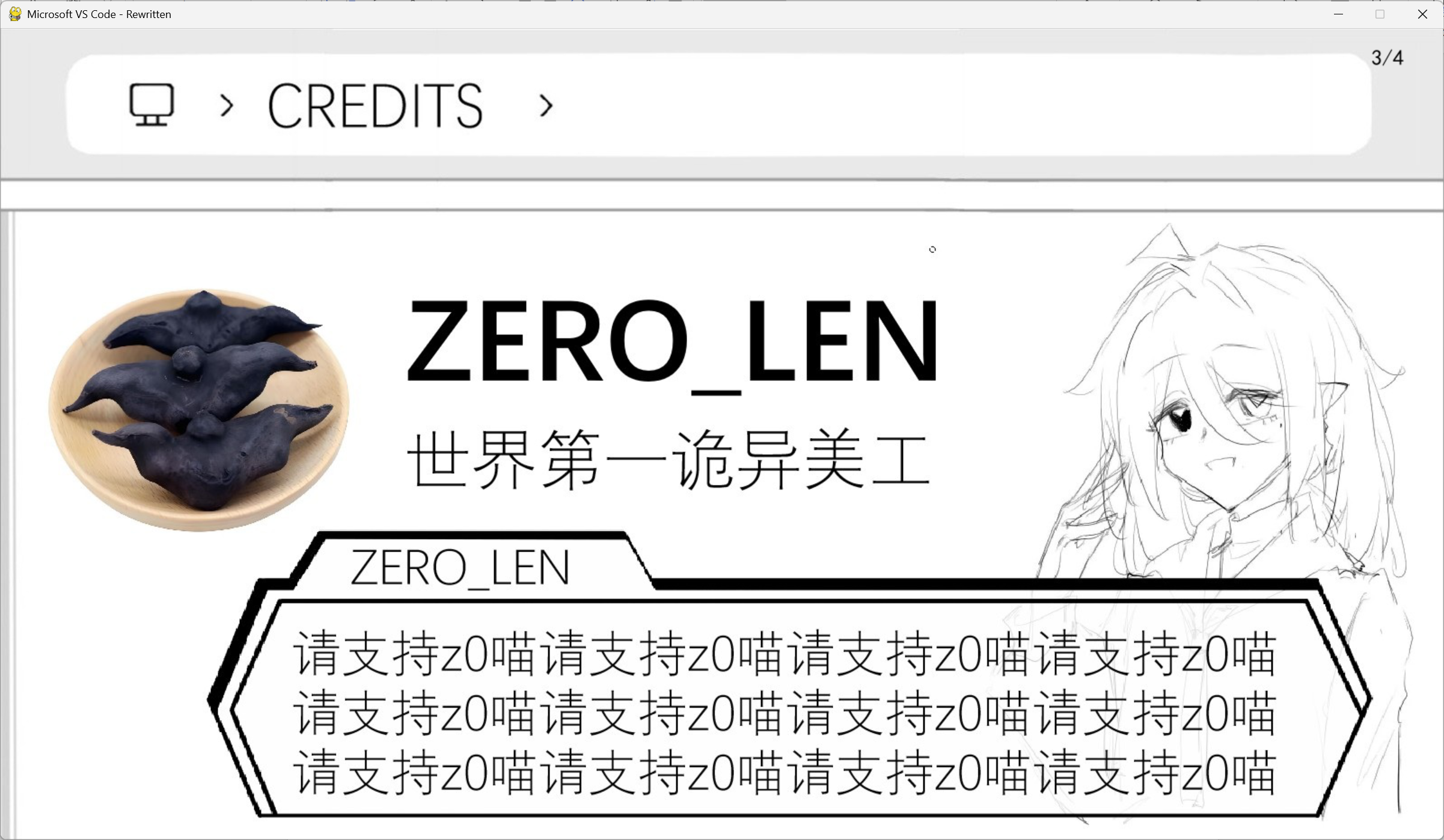


点击三个按钮分别进入对应介绍

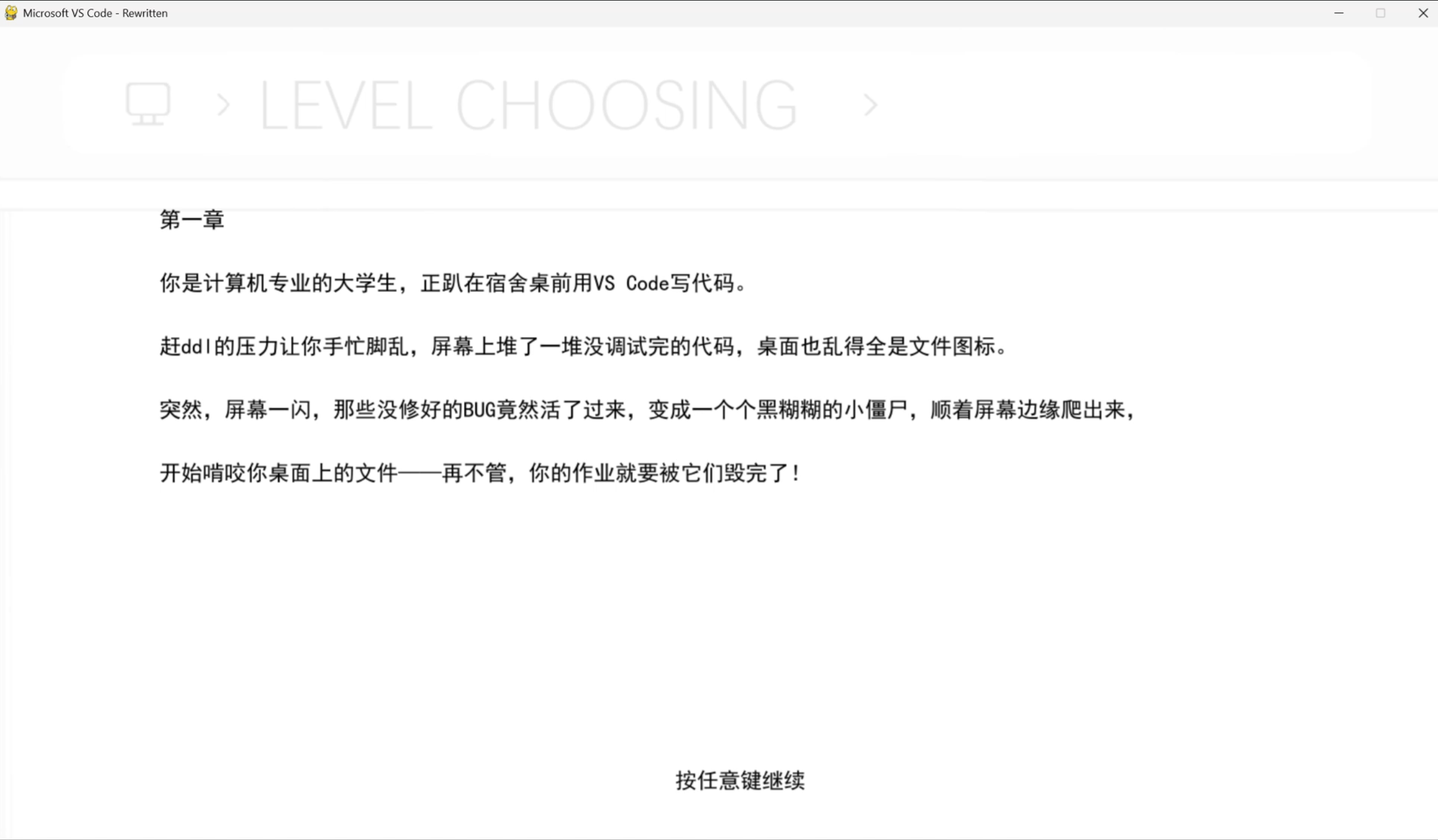
 

CREDITS





Start后点击1-1进入剧情



选择ide



按esc暂停



Lose界面



以下是一些个人想对沙老师说的话

其实与部分同学的评价不同，我挺喜欢沙老师的课，虽然上课跟做作业确实在难度方面有点痛苦，但是不得不说还是学到了东西的。而且沙老师的风格真的挺不一样的（跟国内的老师相比），感谢沙老师让我们做这个游戏，给我这么一个锻炼的机会。最后跟沙老师说goodbye吧，再次感谢沙老师。 ——李福洲