

浙江大学

程序设计专题

大程序报告



大程名称: 玛纳霏的不可思议迷宫

小组成员:

1. 姓名 : ***** 学号: ***** 电话: *****

指导老师: 刘新国

2022~2023 春夏学期 2023 年 4 月 11 日

目 录

| | |
|--|----|
| 1. 大程序简介 | 5 |
| 1.1. 选题背景及意义 | 5 |
| 1.2. 目标要求 | 5 |
| 1.3. 术语说明 | 6 |
| 2. 需求分析 | 8 |
| 2.1. 功能需求 | 8 |
| 2.2. 数据需求 | 9 |
| 2.3. 性能需求 | 9 |
| 3. 程序开发设计 | 9 |
| 3.1. 总体架构设计 | 9 |
| 3.2. 功能模块设计 | 11 |
| 3.2.1. 状态管理模块 <i>State Manager</i> | 11 |
| 3.2.2. 事件处理模块 <i>Controller</i> | 11 |
| 3.2.3. 页面模块 <i>Pages</i> | 12 |
| 3.2.4. 对话框模块 <i>Dialogs</i> | 16 |
| 3.2.5. 操作模块 <i>Attempt</i> | 17 |
| 3.2.6. 数据库模块 <i>Database</i> | 17 |
| 3.2.7. 迷宫模块 <i>Dungeon</i> | 18 |
| 3.2.8. 算法模块 <i>Algorithm</i> | 18 |
| 3.2.9. 绘图模块 <i>Graphics</i> | 19 |
| 3.2.10. 杂项模块 <i>Utils</i> | 20 |
| 3.3. 数据结构设计 | 21 |
| 3.3.1. <i>utils.h</i> | 21 |
| 3.3.2. <i>utils.c</i> | 22 |
| 3.3.3. <i>appstate.h</i> | 23 |
| 3.3.4. <i>statemanager.c</i> | 23 |
| 3.3.5. <i>controller.h</i> | 23 |
| 3.3.6. <i>controller.c</i> | 24 |
| 3.3.7. <i>pokemon.h</i> | 24 |
| 3.3.8. <i>items.h</i> | 26 |
| 3.3.9. <i>landevents.h</i> | 27 |
| 3.3.10. <i>dungeon.h</i> | 27 |
| 3.3.11. <i>itembag.h</i> | 28 |
| 3.3.12. <i>solvemodel.h</i> | 28 |
| 3.3.13. <i>solvemodel.c</i> | 28 |
| 3.3.14. <i>helplist.h</i> | 29 |
| 3.3.15. <i>helplist.c</i> | 29 |
| 3.3.16. <i>globalvalue.h</i> | 29 |
| 3.3.17. <i>attempt.h</i> | 29 |
| 3.3.18. <i>messagedialog.h</i> | 30 |

| | |
|--|-----|
| 3.3.19. <i>getfilename dialog.h</i> | 30 |
| 3.3.20. <i>editpage.h</i> | 30 |
| 3.3.21. <i>editpage.c</i> | 30 |
| 3.3.22. <i>simage.h</i> | 31 |
| 3.3.23. <i>simage.c</i> | 31 |
| 3.3.24. <i>enemylst.h</i> | 32 |
| 3.3.25. <i>explorer.h</i> | 32 |
| 3.3.26. <i>explorers.c</i> | 32 |
| 3.3.27. <i>helppage.h</i> | 32 |
| 3.3.28. <i>AppStates</i> | 33 |
| 3.4. 函数设计描述 | 34 |
| 3.4.1. <i>aboutpage (.c, .h)</i> | 34 |
| 3.4.2. <i>alertdialog (.c, .h)</i> | 36 |
| 3.4.3. <i>attempt (.c, .h)</i> | 37 |
| 3.4.4. <i>confirm dialog (.c, .h)</i> | 38 |
| 3.4.5. <i>controller (.c, .h)</i> | 40 |
| 3.4.6. <i>drawdungeon (.c, .h)</i> | 42 |
| 3.4.7. <i>drawitembag (.c, .h)</i> | 46 |
| 3.4.8. <i>dungeon (.c, .h)</i> | 47 |
| 3.4.9. <i>dungeonprocess (.c, .h)</i> | 51 |
| 3.4.10. <i>editpage (.c, .h)</i> | 52 |
| 3.4.11. <i>enemylst (.c, .h)</i> | 58 |
| 3.4.12. <i>explorer (.c, .h)</i> | 60 |
| 3.4.13. <i>getfilename dialog (.c, .h)</i> | 66 |
| 3.4.14. <i>helplist (.c, .h)</i> | 67 |
| 3.4.15. <i>helppage (.c, .h)</i> | 68 |
| 3.4.16. <i>imagesupport (.c, .h)</i> | 70 |
| 3.4.17. <i>itembag (.c, .h)</i> | 70 |
| 3.4.18. <i>items (.c, .h)</i> | 72 |
| 3.4.19. <i>landevent (.c, .h)</i> | 73 |
| 3.4.20. <i>main (.c)</i> | 73 |
| 3.4.21. <i>messagedialog (.c, .h)</i> | 74 |
| 3.4.22. <i>pages (.c, .h)</i> | 75 |
| 3.4.23. <i>pausepage (.c, .h)</i> | 81 |
| 3.4.24. <i>pokemon (.c, .h)</i> | 82 |
| 3.4.25. <i>simage (.c, .h)</i> | 85 |
| 3.4.26. <i>solvemodel (.c, .h)</i> | 89 |
| 3.4.27. <i>statemanager (.c, .h)</i> | 95 |
| 3.4.28. <i>statusbar (.c, .h)</i> | 100 |
| 3.4.29. <i>utils (.c, .h)</i> | 100 |
| 3.5. 源代码文件组织设计 | 101 |
| 3.5.1. 文件目录结构 | 101 |
| 3.5.2. 文件函数结构 | 102 |
| 3.5.3. 多文件构成机制 | 103 |

| | |
|---------------------------|------------|
| 4. 部署运行和使用说明 | 103 |
| 4.1. 编译安装 | 103 |
| 4.1.1. 使用 Dev C++ | 103 |
| 4.2. 运行测试 | 104 |
| 4.3. 用户使用手册 | 107 |
| 5. 团队合作 | 112 |
| 5.1. 开发日志 | 112 |
| 5.2. 编码规范 | 112 |
| 5.3. 任务分工 | 112 |
| 5.4. 个人遇到的难点与解决方案 | 113 |
| 5.5. 合作总结 | 113 |
| 5.5.1. 开发亮点 | 113 |
| 5.5.2. 开发挑战点 | 113 |
| 5.5.3. 应用知识点 | 113 |
| 5.5.4. 合作记录 | 114 |
| 5.6. 收获感言 | 114 |
| 6. 参考文献资料 | 115 |

《玛纳霏的不可思议迷宫》大程序设计项目

1. 大程序简介

1.1. 选题背景及意义

“不可思议迷宫”是一种电子游戏的要素。在“不可思议迷宫”中，玩家需要在随机生成的迷宫中生存，收集有利的道具、打败强大的敌人。

玛纳霏是《宝可梦》系列的一个角色。作为一个活泼而乐观的小精灵，他非常喜欢探索未知、寻找宝藏。为了锻炼他的探险能力，我们将为他开发一个用于编辑和运行不可思议迷宫的程序，让他可以在模拟训练下磨练自己的探险策略。



1.2. 目标要求

设计一个迷宫游戏的地图编辑器，并实现游戏的手动和程序求解。

1. 基本功能要求：

(1) 地图生成：地图中至少需包含玩家、障碍物、起点、终点等元素。
支持交互式手动编辑地图和自动随机生成地图。

(2) 迷宫求解：手动求解，使用键盘方向键走迷宫，到达终点即胜利；
程序自动求解，可视化显示最短路线、所有可行路线等，支持单步执行、自动执行等。

(3) 支持菜单和工具栏（快捷键）功能：

- ① 文件：新建地图、打开地图、保存地图、退出等，
- ② 地图编辑：随机生成、手动编辑、元素选择等
- ③ 地图求解：手动求解、程序求解、展示所有可行解等
- ④ 帮助：使用说明、关于

2. 较高功能要求：

(1) 手动求解的提示功能：提示下一步可走的格子。

(2) 程序求解过程可视化：除了可视化最短路径，可以将求解的过程可视化。

(3) 增强游戏性，给玩家添加生命值属性，增加门/钥匙道具，消耗钥匙可以打开门，在此基础上找到最优路径。

1.3. 术语说明

宝可梦（Pokémon）：一种既神秘又不可思议的生物。宝可梦可以学会各种招式，使用招式可以对战或进行各种活动。

不可思议迷宫（Mystery Dungeon）：一种包含楼梯（入口、出口等）、陷阱、道具等各种因素的迷宫，较一般的迷宫更为复杂。不可思议迷宫可看作是由格子组成的方阵。

探险队：一种冒险团体，主要目标有探索不可思议迷宫等。在本案例中，只考虑由一个宝可梦组成的探险队。

玛纳霏（Manaphy）：《宝可梦》系列的一个角色。

克雷色利亚（Cresselia）：《宝可梦》系列的一个角色，见右图。她是一个善良而富有经验的强大宝可梦。



宝可梦相关术语

HP (Hit Point)：衡量宝可梦的体力。HP 越高，代表宝可梦的体力越充沛、活力越强。受到伤害时，HP 会相应地减少。当某个宝可梦的 HP 减少到 0，那么他/她将倒下，无法继续探险。

饱腹度（Belly）：衡量宝可梦的饱腹度。在探险过程中会逐渐感受到饿，相当于饱腹度降低。当某个宝可梦的饱腹度减少到 0，那么他/她的 HP 将会快速下降，直至力尽倒下。

等级（Level, LV）：表示宝可梦强弱的值。等级越高，宝可梦的能力会越

高。

经验值（Experience, EXP）：衡量宝可梦积累的经验。累积到一定值的时候，宝可梦的等级会提升。

攻击（Attack）：在其他因素相同时，宝可梦的攻击越高，造成的伤害会越多。

防御（Defense）：在其他因素相同时，宝可梦的防御越高，受到的伤害会越少。

招式（Move）：是宝可梦可以使用的特殊能力。在本案例中，招式主要用于给敌人造成伤害。

效果（Effect）：在其他因素相同时，招式的效果越高，造成的伤害会越高。

PP 值（Power Point, PP）：宝可梦使用某个招式时需要的值。某个招式的 PP 值耗尽的话，将不能再使用。

不可思议迷宫相关术语

起点（Start）：我方宝可梦进入不可思议迷宫时，一开始处于的位置。

终点（End / Destination）：不可思议迷宫的出口。

平地（Plain）：可供行走的位置。

障碍（Block）：不能走过的位置。

场地事件（Land Event）：当宝可梦踩上某个格子的时候，将会触发的事件。

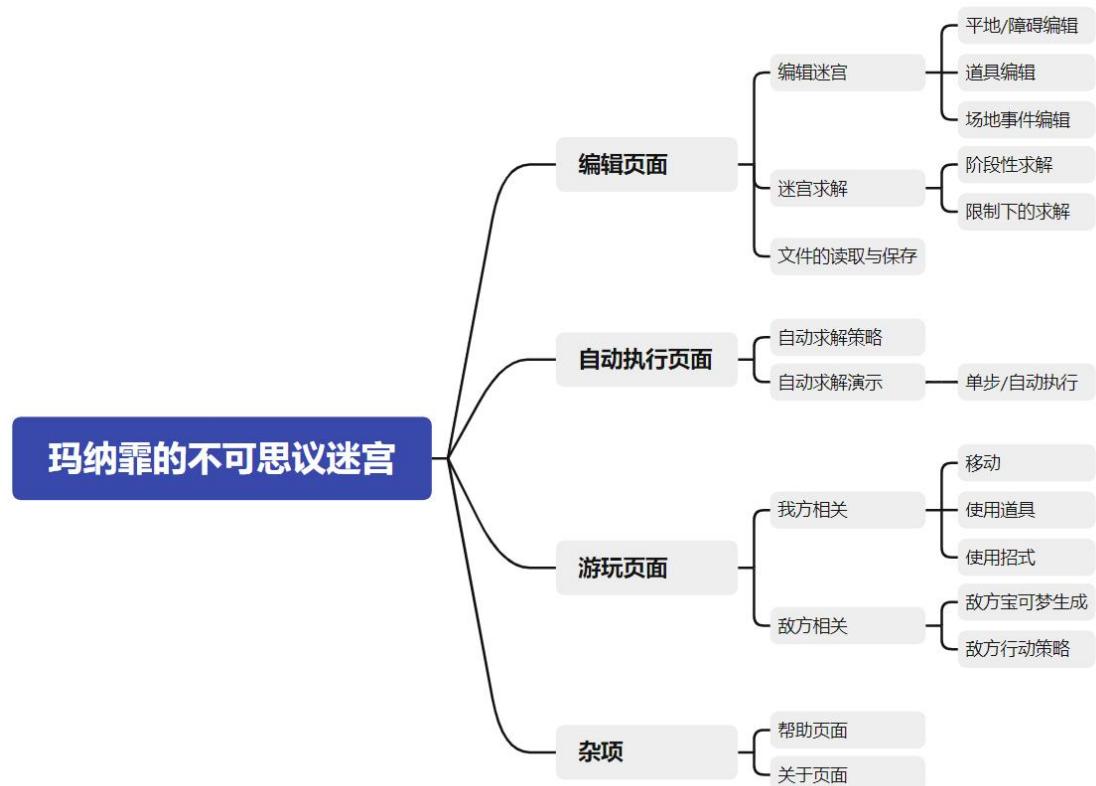
道具（Item）：宝可梦可以通过使用道具回复 HP 或饱腹度、学会新招式等。道具可以在不可思议迷宫里生成。

道具背包（Item Bag）：我方宝可梦可以同时持有很多个道具，它们都放在道具背包里，可以随时取用。

具体道具和场地事件请参阅编辑器的帮助页面。

2. 需求分析

2.1. 功能需求



本程序是一个不可思议迷宫编辑器，附带有游玩不可思议迷宫的功能。其中主要实现的页面分别是编辑页面、自动执行页面和游玩页面。

1. 编辑页面

这个页面的主要功能为编辑不可思议迷宫。需要至少支持迷宫的大小修改、迷宫格子平地/障碍状态修改、格子道具编辑和场地事件的编辑，以及随机生成迷宫。另外还需要有从文件读取迷宫，与保存迷宫到文件的功能。

另外，为了改善用户体验，这个页面还可以提供缩放迷宫的功能，以及在这个页面查看当前迷宫的解的功能，以及求解的过程可视化。

2. 自动执行页面

这个页面的主要功能为演示迷宫的解法。需要至少支持求解的单步执行和自动执行。另外可以支持解法路线的记录。

3. 游玩页面

这个页面的主要功能为让用户（玩家）自己（以玛纳霏的身份）游玩当前迷宫。其中，玩家可以进行的操作有移动、使用道具和使用招式；玩家可以选择性地生成敌人，敌人会根据一定策略移动和攻击玩家。另外可以支持提示下一步可走的格子。

除了这三个页面外，本程序还需要实现帮助页面和关于页面。帮助页面主要用于解释迷宫内元素的含义，关于页面主要用于版权的声明。

2.2. 数据需求

1. 本程序需要支持从文件中读取迷宫数据，和将迷宫数据保存到文件。
2. 迷宫作为多个二维数组（格子状态、道具、场地事件等）进行存储。
3. 本程序还需要读取众多 `bmp` 格式贴图，用于绘制图像。出于性能考虑，贴图只会在程序开始运行时读取一次，之后在内存中调用。

2.3. 性能需求

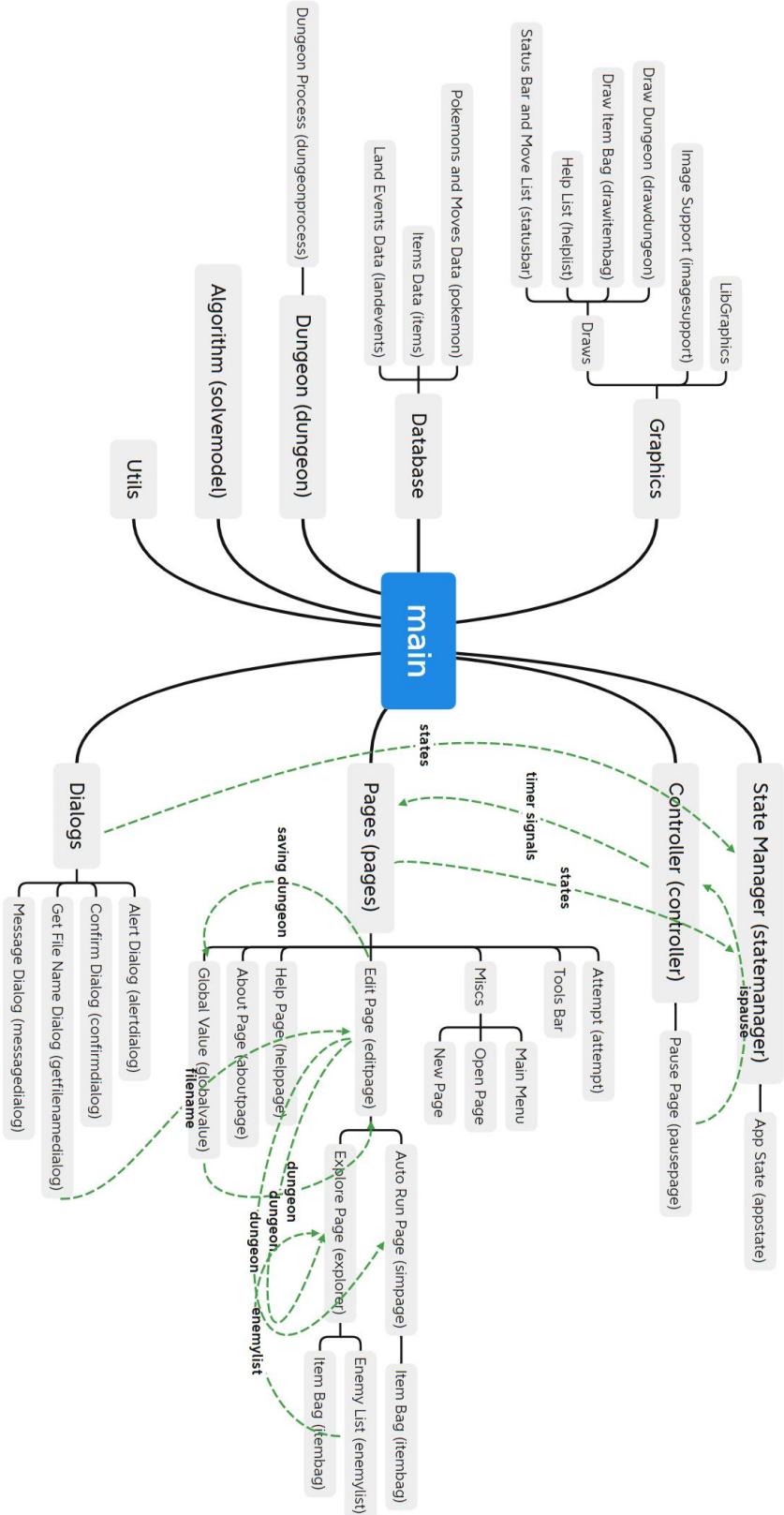
1. 本程序的目标帧率为 100 帧每秒，单帧的渲染时间仅有 10 毫秒。
2. 因为有将二维的迷宫绘制到屏幕上的需求，因此有至少 $O(n^2)$ 的时间复杂度（ n 为迷宫的长或宽，最多为 50）。
3. 寻找迷宫的解所使用的堆优化状态压缩迪科斯彻算法的时间复杂度约为 $O((n^2 2^k) \log(n^2 2^k))$ ， k 为钥匙的种类数，最多为 4。（从表达式可以看出， k 对算法复杂度的影响巨大，因此 k 的值无法很大。）

3. 程序开发设计

3.1. 总体架构设计

下图介绍了这个程序的功能模块架构。全局模块包括状态管理模块 `StateManager` 和事件处理模块 `Controller`；业务模块包含页面模块 `Pages`、对话框模块 `Dialogs` 和操作模块 `Attempt`；绘图模块包含 `LibGraphics`、图

片支持模块 **Image Support** 和具体的绘制模块 **Draws**; 数据模块包含数据库模块 **Database** 和迷宫模块 **Dungeon**; 以及算法模块 **Algorithm**。



3.2. 功能模块设计

3.2.1. 状态管理模块 State Manager

在这个程序中，所有的页面（Pages）和一部分对话框（Dialogs）都被视为一种“状态”（State）。程序在不同的状态之间转移，执行对应状态的工作。

```
typedef struct AppState {
    int uid;

    voidFn ctor;
    voidFn proc;
    voidFn dtor;

    keyboardCallback fnKey;
    charCallback fnChar;
    mouseCallback fnMouse;
} AppState;
```

状态有它的构造函数（ctor）、运行函数（proc）和析构函数（dtor）；以及自己处理各种回调函数的方法。状态所需要的绘图过程一般都在运行函数中。

```
typedef struct StateStack {
    AppState *stk[MaxStateNumber];
    size_t top;
} StateStack;
```

```
StateStack stateStack;
```

而整个程序的当前状态由一个栈来维护。后面入栈的状态的某些数据，会来自于先前入栈的状态；而先出栈的状态可以传递参数给后出栈的状态。状态管理模块在一般情况下只会运行当前栈顶的运行函数。比如，一个页面传递一个默认文件名给一个询问文件名的对话框；对话框状态压在页面上，必须等对话框执行完页面才能继续执行；对话框状态结束后，页面就能拿取对话框的文件名了。

3.2.2. 事件处理模块 Controller

事件处理模块用于统一处理大部分通用的键盘事件，以及一些定时器事件。常见的定时事件有重绘页面，以及长按方向键导致的连续移动等。

3.2.3. 页面模块 Pages

页面模块包含了这个程序中的所有的页面；同时也包含工具栏的绘制。

3.2.3.1. 工具栏 Tools Bar

工具栏要根据当前程序的状态来绘制不同的选项。比如当没有打开一个迷宫文件的时候，程序就不能“运行”迷宫而只能打开迷宫。

3.2.3.2. 主页面 Main Menu



程序的主页面。

通过工具栏来进入各个页面。

3.2.3.3. 打开页面和新建页面 Open Page, New Page



这两个页面接受一个字符串，作为文件名。虽然看起来是个对话框，但是它们的层级依然是页面，只是这个页面只包含这个对话框。

3.2.3.4. 编辑页面 Edit Page



中间为迷宫视图；左上角为操作指引；左侧中部为求解模块的前端；左下角为改变迷宫大小与改变视图缩放；右上角为编辑模式；右下角为迷宫的道具和场地事件编辑。

3.2.3.5. 自动执行页面 Auto Run Page



在自动执行页面中，会请克雷色利亚来让她尝试自动完成迷宫。

中间为迷宫视图；左上角为操作指引；左侧中上部为提升克雷色利亚等级和恢复克雷色利亚的选项；中部为自动演示相关的选项；左下角为克雷色利亚的状态栏；右上角为她的道具背包；右下角为她的招式列表。

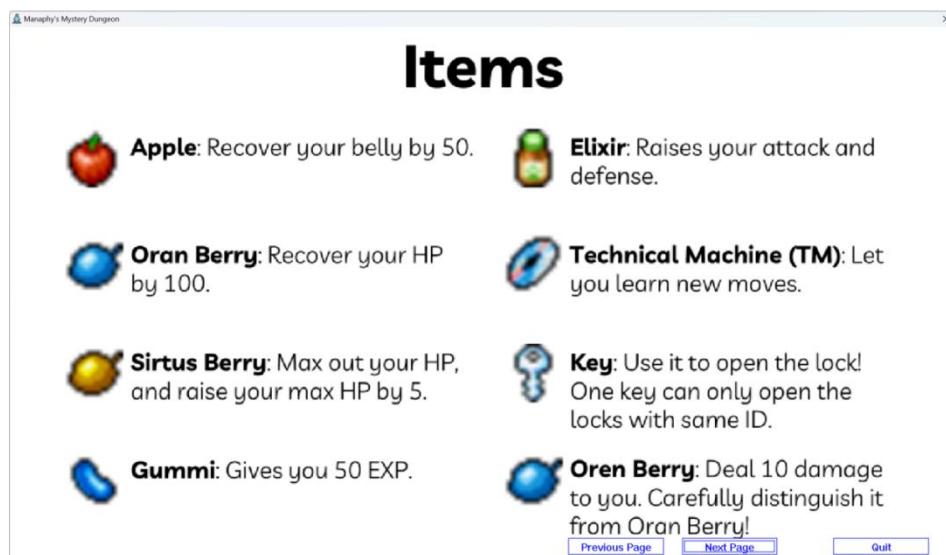
3.2.3.6. 游玩页面 Explore Page



在游玩页面中，你将以玛纳霏的身份完成这个迷宫。注意在这个期间视角和缩放都是固定的。

中间为迷宫视图；左上角为操作指引；左侧中部为生成敌人相关的选项；左下角为玛纳霏的状态栏；右上角为道具背包；右下角为招式列表。另外在工具栏中有作弊选项。

3.2.3.7. 帮助页面 Help Page



通过工具栏进入帮助页面。帮助页面需要绘制指定的帮助图像。通过按键来翻页和退出帮助页面。

3.2.3.8. 关于页面 About Page



通过工具栏进入关于页面。关于页面包含了本项目名字、作者与致谢名单。

通过按键退出关于页面。（iotang 和 Kate da Explorers of Sky 是同一个人，这两个都是作者的别名。）

3.2.4. 对话框模块 Dialogs

对话框模块包含了这个程序中的所有的对话框。一部分的对话框拥有页面的等级。

3.2.4.1. 警告对话框 Alert Dialog



警告对话框接收多个参数以适应不同的使用场景。这个对话框用于提示用户一些关键消息。

3.2.4.2. 确认对话框 Confirm Dialog



确认对话框接收多个参数以适应不同的使用场景。这个对话框用于提示用户接下来操作的风险。

3.2.4.3. 获取文件名对话框 Get File Name Dialog



获取文件名对话框让用户可以输入一个字符串，这个字符串作为文件名。

3.2.4.4. 消息对话框 Message Dialog



消息对话框展示了游玩过程中的消息提示。它会选择最近的四条消息来展示。

3.2.5. 操作模块 Attempt

操作模块规定了不同操作的编码和解码方式。

3.2.6. 数据库模块 Database

数据库模块存储了程序运行所需要的数据。

3.2.6.1. 宝可梦数据模块 Pokemons and Moves Data

这个模块存储了宝可梦的数据和宝可梦的招式的数据。

3.2.6.2. 道具数据模块 Items Data

这个模块存储了道具的数据。

3.2.6.3. 场地事件数据模块 Land Events Data

这个模块存储了场地事件的数据。

3.2.7. 迷宫模块 Dungeon

这个模块定义了迷宫的数据结构，以及有关迷宫的数据处理方法。

3.2.7.1. 迷宫进程模块 Dungeon Process

这个模块定义了涉及迷宫的操作，比如宝可梦捡拾迷宫的道具、触发格子的场地事件等。

3.2.8. 算法模块 Algorithm

算法模块主要用于寻路。算法模块所使用的算法是状态压缩堆优化迪科斯彻算法。

3.2.8.1. 关于状态压缩堆优化迪科斯彻算法

宝可梦根据他们持有的钥匙种类的状态，可以在迷宫中走到不同的范围。如果钥匙种类有 K 种，那么根据每一种钥匙有或者没有的情况，可以划分为 2^K 种状态。

对于迷宫的每个格子，我们创建 2^K 个结点，代表一个宝可梦位于这个格子上，他/她所持有的钥匙状态为某个值。

相邻的格子的结点之间会有连边。如果结点所属的格子有钥匙，那么这个结点所到达的结点的所属钥匙状态也会改变。如果结点所属的格子有锁，那么只有那些有对应钥匙的结点可以连入边。

如果结点所属的格子有会导致 HP 损失的场地事件，那么走到这个结点会付出额外的距离代价，取决于预先传入的参数。

3.2.9. 绘图模块 Graphics

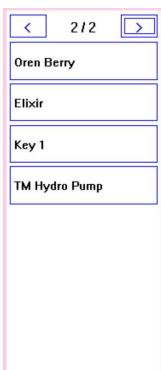
绘图模块种有一些模块化的绘制函数。

3.2.9.1. 迷宫绘图模块 Draw Dungeon

这个模块可以根据给定的参数绘制迷宫。

3.2.9.2. 道具背包绘图模块 Draw Item Bag

这个模块可以绘制道具背包，并返回道具的使用/丢弃指令。



3.2.9.3. 帮助栏模块 Help List

这个模块可以绘制帮助栏。

```

Move: Arrow or WASD
Rest: Space
Change Direction:
Shift-Arrow or Shift-WASD
Use Items: Click on list
Use Moves: Number 1-5
Drop Items:
Ctrl and Right-click on list
Unlearn Moves:
Ctrl and Right-click on list

```

3.2.9.4. 状态栏和招式栏模块 Status Bar and Move List



这个模块可以绘制宝可梦的状态栏和宝可梦的招式栏。

左图是等级 80 的克雷色利亚的状态栏，包含她的头像、名字、等级、经验条、HP 条、饱腹度、攻击和防御。

右图是玛纳霏学会水炮 (Hydro Pump) 后的招式栏，包含招式名字、效果和剩余 PP 值。招式栏可以返回招式的使用/遗忘指令。

| | |
|------------|-----------|
| Tackle | |
| Eff. 20 | Unlimited |
| Water Gun | |
| Eff. 40 | 25 / 25 |
| Hydro Pump | |
| Eff. 100 | 5 / 5 |

3.2.9.5. 图像支持模块 Image Support

图像支持模块提供了对图像读取的支持。对图像的绘制函数是在 `LibGraphics` 中完善的。

3.2.10. 杂项模块 Utils

杂项模块提供了一些常量和时常使用的函数的支持，比如方向与对应 x 轴 / y 轴增减的变化。

3.3. 数据结构设计

3.3.1. utils.h

```
#define ProjectName ("Manaphy's Mystery Dungeon") // 这个项目的全名。  
  
#define FrameLength (10) // 每一帧的时间长度。  
  
#define WindowWidthInch (16.00) // 窗口的宽度。  
#define Window43Gap (2.00) // 16:9 相比 4:3 左右分别多出来的长度。  
#define Window43Left (2.00) // 4:3 相对 16:9 的宽度的开始位置。  
#define Window43Right (14.00) // 4:3 相对 16:9 的宽度的结束位置。  
#define Window43Width (12.00) // 4:3 下窗口的宽度。  
#define WindowHeightInch (9.00) // 窗口的高度。  
  
#define MenuHeight (WindowHeightInch * 0.03) // 工具栏等栏目的高度。  
  
#define MaxFileNameLength (20) // 文件名长度的最大值。  
  
typedef long long lint;  
  
typedef enum {  
    idMainMenu,  
    idHelpPage,  
    idAboutPage,  
    idAlertDialog,  
    idConfirmDialog,  
    idNewPage,  
    idOpenPage,  
    idSaveAsPage,  
    idEditPage,  
    idSimPage,  
    idExplorer,  
    idPausePage  
} AppStateID; // 状态 ID。
```

```

typedef enum TimeEvent {
    ScreenRend,
    FaceRight,
    FaceUp,
    FaceLeft,
    FaceDown,
    MoveRight,
    MoveUp,
    MoveLeft,
    MoveDown,
    MoveNoDirection,
    UseMove1,
    UseMove2,
    UseMove3,
    UseMove4,
    UseMove5,
    MessageExpire,
    AutoRun,
    ClearUsingMove,
    PauseBufferExpire
} TimeEvent; // 计时器事件 ID。

typedef enum { Player, Enemy } Role; // 角色的阵营。
typedef enum {
    RIGHT,
    UP,
    LEFT,
    DOWN,
    NODIRECTION,
    ERRORDIRECTION
} Direction; // 朝向。

extern int go[5][2]; // 朝向与坐标的对应换算。

#define inf (0x3f3f3f3f) // int 下的无穷大，可以被 memset 出来。
#define linf (0x3f3f3f3f3f3f3fll) // long long 下的无穷大。

```

3.3.2. utils.c

```

int go[5][2] = {
    {1, 0}, {0, 1}, {-1, 0}, {0, -1}, {0, 0}}; // 朝向与坐标的
对应换算。

```

3.3.3. appstate.h

```

typedef void (*voidFn)(void);
typedef void (*keyboardCallback)(int button, int event); // 键盘回调函数。
typedef void (*charCallback)(int ch); // 输入字符回调函数。
typedef void (*mouseCallback)(int x, int y, int button,
                             int event); // 鼠标回调函数。

typedef struct AppState {
    int uid; // 用于区分不同的状态。

    voidFn ctor; // 状态的构造函数。
    voidFn proc; // 状态的运行函数。
    voidFn dtor; // 状态的析构函数。

    keyboardCallback fnKey; // 状态的键盘回调函数。
    charCallback fnChar; // 状态的输入字符回调函数。
    mouseCallback fnMouse; // 状态的鼠标回调函数。
} AppState;

```

3.3.4. statemanager.c

```

#define MaxStateCount (99) // 最大的状态堆栈数量。（实际上连 10 都不会超过。）

typedef struct StateStack {
    AppState *stk[MaxStateCount];
    size_t top;
} StateStack; // 用于存放状态的栈数据结构。

StateStack stateStack; // 用于存放状态的栈。对于其他模块不可见。
void (*currentStateProc)(void); // 当前状态的运行函数。

```

3.3.5. controller.h

```

#define MoveGap (150) // 长按移动键时，每隔多少毫秒移动一次。
#define UseMoveGap (500) // 长按招式键时，每隔多少毫秒使用一次招式。并未采用。

extern int controlPressed; // 是否按下了 Control。

```

```
extern int shiftPressed; // 是否按下了 Shift。
```

3.3.6. controller.c

```
extern void (*currentStateProc)(void); // 当前状态的运行函数。  
在每一帧调用。
```

```
int timerStarted[99] = {0}; // 定时器是否已经启动。  
int isUsingMove = 0; // 是否有招式键被按下。  
int pauseBuffer = 0; // 用于优化暂停下的回调阻塞。克雷色利亚教我写的
```

```
int controlPressed = 0; // 是否按下了 Control。  
int shiftPressed = 0; // 是否按下了 Shift。
```

3.3.7. pokemon.h

```
#define MaxMoveCount (5) // 一个宝可梦最多能学会的招式数量。  
#define MaxPokemonNameLength (16) // 宝可梦名字的最大长度。  
#define MaxMoveNameLength (16) // 招式名字的最大长度。  
  
#define PokemonSpeciesNumber (1011) // 宝可梦图鉴的最大编号。  
// 宝可梦图鉴编号。  
#define NKate (0)  
#define NManaphy (490)  
#define NCresselia (488)  
#define NRemoraid (223)  
#define NSuicune (245)  
  
#define MaxMoveNumber (5) // 招式的最大编号。  
// 招式的编号。  
#define MTackle (0)  
#define MWaterGun (1)  
#define MBubbleBeam (2)  
#define MSurf (3)  
#define MHydroPump (4)  
#define MPsychic (5)  
  
typedef struct Move {  
    char name[MaxMoveNameLength + 1];  
    int effect, pp;
```

```

} Move; // 招式。

typedef enum Gender { Female, Male } Gender; // 性别（或性别认同）。

typedef struct Pokemon {
    Role role;

    char name[MaxPokemonNameLength + 1];
    Gender gender;
    int species;
    int lv;
    double exp;

    double belly, maxbelly;

    int hp, maxhp;
    int atk, def;

    int moveCount;
    int move[MaxMoveCount], pp[MaxMoveCount];

    int x, y; // 在迷宫中的位置。
    Direction direction; // 朝向。
} Pokemon; // 宝可梦的数据结构。

typedef struct Pokedex {
    char name[MaxPokemonNameLength + 1];
    double hpBase;
    double atkBase, defBase;
    double hpGrowth;
    double atkGrowth, defGrowth;

    HBITMAP sprites[4]; // 在迷宫中的形象。
    HBITMAP portrait; // 头像。
} Pokedex; // 宝可梦的对应种族的基础数据，即宝可梦图鉴。

extern Pokedex pokedex[PokemonSpeciesNumber + 1]; // 宝可梦图鉴。

```

```
extern Move movedex[MaxMoveNumber + 1]; // 招式图鉴。
```

3.3.8. items.h

```
#define MaxItemNumber (8) // 道具编号的最大值。
#define MaxItemNameLength (16) // 道具名字长度最大值。
```

// 道具编号定义。

```
#define INone (0)
#define IApple (1)
#define IGummi (2)
#define IOranBerry (3)
#define IOrenBerry (4)
#define ISitrusBerry (5)
#define IElixir (6)
#define IKey (7)
#define ITM (8)
```

```
#define MaxKeyID (4) // 钥匙的最大编号。钥匙的编号从 1 开始。
```

```
typedef struct ItemData {
    char name[MaxItemNameLength + 1];
    double dexp;
    double dbelly;
    int dhp, dmaxhp;
    int datk, ddef;
    int dmove; // 可以学会的招式编号。
    int keyid; // 作为钥匙的编号。
    int defaultArg;
    HBITMAP sprite;
} ItemData; // 道具数据。
```

```
typedef struct Item {
    int type;
    int arg;
} Item; // 具体的道具。
```

```
extern ItemData itemsData[MaxItemNumber + 1]; // 道具图鉴。
```

3.3.9. landevents.h

```
#define MaxLandEventTypeNumber (4) // 最大场地事件编号。
#define MaxLandEventNameLength (16) // 场地事件名字的最大长度。

typedef enum LandEventType {
    None,
    Lock,
    Damage,
    DamageOT,
    HealOT
} LandEventType; // 场地事件编号定义。

typedef struct LandEventData {
    char name[MaxLandEventNameLength + 1];
    int permanent;
    int defaultArg;
    HBITMAP sprite;
} LandEventData; // 场地事件数据。

typedef struct LandEvent {
    int type;
    int arg;
} LandEvent; // 具体的场地事件。

extern LandEventData
landEventsData[MaxLandEventTypeNumber + 1]; // 场地事件图
鉴。
```

3.3.10. dungeon.h

```
#define MaxDungeonWidth (50) // 最大的迷宫宽度。
#define MaxDungeonHeight (50) // 最大的迷宫高度。

typedef enum LandType { Plain, Block, Start, End } LandType;
// 迷宫格子类型。

typedef struct Dungeon {
    int width, height;
    int mp[MaxDungeonWidth][MaxDungeonHeight];
```

```

LandEvent event[MaxDungeonWidth][MaxDungeonHeight]; // 场地事件。
Item item[MaxDungeonWidth][MaxDungeonHeight]; // 道具。
} Dungeon;

```

3.3.11. itembag.h

```
#define MaxItemBagVolume (2000) // 最多持有的道具数量。
```

```

typedef struct ItemBag {
    size_t count; // 道具数量。
    Item items[MaxItemBagVolume]; // 道具数据。
    int currentPage; // 现在在道具栏的哪一页。
} ItemBag;

```

3.3.12. solvemodel.h

```

#define SolveStateCount (40007) // 求解状态的数量限制，至少要有 n^2 2^k。
#define DefaultHPPenalty (10000000ll) // 默认 HP 损失惩罚系数。

```

```

typedef struct RouteNode {
    int x, y;
    struct RouteNode *nex;
} RouteNode; // 用于存储迷宫解路线的链表的结点。

```

```

typedef struct DungeonSolution {
    int mp[MaxDungeonWidth][MaxDungeonHeight]; // 是否到达过 / 已经到达。
    int routeValid; // 迷宫解路线是否可用。
    RouteNode *route; // 迷宫解路线的链表的头结点，从起点到终点。
} DungeonSolution; // 用于存储迷宫解的数据结构。

```

3.3.13. solvemodel.c

```
#define SolveHeapSize (200000) // 用于跑迪科斯彻的堆的大小。
```

```

typedef struct HeapNode {
    int stat;
    lint dis;
} HeapNode; // 堆的结点。

```

```
HeapNode heapBuf[SolveHeapSize + 1]; // 堆的结点集合。
HeapNode infNode = {inf, linf};
int heapSize; // 堆的目前大小。
```

3.3.14. helpList.h

```
#define MaxHelpListLength (20) // 帮助栏最长条目数。
```

3.3.15. helpList.c

```
typedef struct HelpEntry {
    char *fun; // 左侧的字符串。
    char *key; // 右侧的字符串。
} HelpEntry; // 帮助栏的一个条目。

HelpEntry helpList[MaxHelpListLength]; // 帮助栏的条目。
int helpListLength; // 帮助栏的条目数量。
```

3.3.16. globalvalue.h

```
extern Dungeon
    currentDungeon; // 当前文件的迷宫。只能和编辑页面之间进行交互，只在读取文件和保存文件的时候更改。
extern int isDungeonOpened; // 当前是否打开了迷宫。
extern int
    isDungeonGameOver; // 游玩页面中，游玩是否结束（走到终点或者在迷宫中倒下）。
extern int
    isDungeonSimTerminated; // 自动执行页面中，执行是否结束（克雷色利亚走到终点或者在迷宫中倒下）。
extern int modifiedSinceLastSave; // 自从上一次保存后，是否对迷宫进行了更改。
```

3.3.17. attempt.h

```
#define PFace (10)      // 改变朝向的指令的这一位为 1。
#define PMove (11)       // 移动指令的这一位为 1。
#define PUseItem (12)     // 使用道具指令的这一位为 1。
#define PUseMove (13)     // 使用招式指令的这一位为 1。
#define PRemoveItem (14)   // 丢弃道具指令的这一位为 1。
```

```
#define PRemoveMove (15) // 遗忘招式指令的这一位为 1。
```

3.3.18. messagedialog.h

```
#define MaxMessageLength (200000) // 消息的总长度最大值。
#define MessageExpireTime (3000) // 消息会展示几秒。
#define MaxMessageLine (4) // 一次最多展示消息的数量。
```

3.3.19. getfilenamedialog.h

```
extern char dialogFileName[MaxFileNameLength + 1]; // 读取到的文件名。
```

3.3.20. editpage.h

```
extern char editDungeonFileName[MaxFileNameLength + 1]; // 编辑页面的文件名。
```

```
typedef enumEditMode {
    Targeted,
    SetLandEvent,
    Flip,
    SetPlain,
    SetBlock,
    PlaceStart,
    PlaceEnd
}EditMode; // 编辑模式。
```

3.3.21. editpage.c

```
int editHasReadDungeon; // 是否已经读取了迷宫。
Dungeon editDungeon; // 编辑页面的迷宫，和之前的 currentDungeon 独立。
char editDungeonFileName[MaxFileNameLength + 1]; // 编辑页面的迷宫名字。
```

```
int editHasSolution; // 现在是否有解要显示。
DungeonSolution editDungeonSolution; // 迷宫的解。
lint editDungeonSolutionLimit; // 迷宫的解的限制步数。
lint editDungeonSolutionHPPenalty; // 迷宫的解的 HP 损失惩罚。
```

```

double editCellSize; // 编辑页面的迷宫格子大小。
Pokemon editCamera, editCursor; // 编辑页面的镜头位置和光标位置。
double editMouseX, editMouseY; // 编辑页面的鼠标位置，单位是英寸。
LandEvent editLandEvent; // 编辑页面事件模式下要覆盖成的场地事件。
Item editLandItem; // 编辑页面事件模式下要覆盖成的道具。
int editEventOverrideItem; // 编辑页面事件模式下是否要覆盖场地事件。
int editEventOverrideLandEvent; // 编辑页面事件模式下是否要覆盖道具。

EditMode editMode; // 编辑模式。

int isMouseDownEditPage; // 鼠标是否按下了。
int isJumpedEditPage; // 是否已经使用过右键跳转。

```

3.3.22. simpage.h

```
#define BaseSimulateSpeed (20000.00) // 用于计算克雷色利亚自动演示速度的基准值。
```

3.3.23. simpage.c

```

int simHasReadDungeon; // 是否已经读取了迷宫。
Dungeon simDungeon; // 自动执行页面的迷宫，和之前的 editDungeon 独立。
char simDungeonFileName[MaxFileNameLength + 1]; // 自动执行页面的迷宫名字。
DungeonSolution simHistory; // 记录克雷色利亚走过的位置。

double simCellSize; // 自动执行页面的迷宫格子大小。
Pokemon simCamera, cresselia; // 自动执行页面的镜头位置和克雷色利亚。
ItemBag cresseliaItemBag; // 克雷色利亚的道具背包。
double simMouseX, simMouseY; // 自动执行页面的鼠标位置，单位是英寸。

int isMouseDownSimPage; // 鼠标是否按下了。
int isJumpedSimPage; // 是否已经使用过右键跳转。

int simulateSpeed; // 自动执行的速度。
int isAutoSimulating; // 是否正在自动执行。

```

```
int isCameraFollowCresselia; // 镜头是否锁在克雷色利亚身上。
```

3.3.24. enemylist.h

```
#define MaxEnemyCount (100) // 敌人的数量上限。
```

```
typedef struct EnemyList {
    Pokemon enemy[MaxEnemyCount + 1]; // 敌人的数据。
    Item item[MaxEnemyCount + 1]; // 敌人携带的道具。
    size_t count; // 敌人的数量。
} EnemyList; // 存储敌人列表的数据结构。
```

3.3.25. explorer.h

```
#define runCellSize (1.00) // 游玩时的视角大小。这个值在游玩的时候是锁定的。
```

3.3.26. explorers.c

```
int expHasReadDungeon; // 是否已经读取了迷宫。
Dungeon expDungeon; // 游玩页面的迷宫，和之前的 editDungeon 独立。
char expDungeonFileName[MaxFileNameLength + 1]; // 游玩页面的迷宫名字。
```

```
Pokemon manaphy; // 玛纳霏。
ItemBag manaphyItemBag; // 玛纳霏的道具背包。
```

```
EnemyList enemyList; // 敌人列表。
int isEnemyMove; // 是否处于敌人的回合。
int spawnEnemyCount; // 生成敌人的数量。
int isAutoSpawnEnemy; // 是否随回合自动生成敌人。
```

```
int emptyBellyMessageCount; // 饱腹度空时提示语的播报顺序。
```

3.3.27. helpage.h

```
#define HelpPageCount (4) // 帮助页面的页数。
```

3.3.28. AppStates

```

AppState AboutPage = {idAboutPage,
                     NULL,
                     drawAboutPage,
                     NULL,
                     uiAboutPageGetKeyboard,
                     uiAboutPageGetChar,
                     uiAboutPageGetMouse};

AppState AlertDialog = {idAlertDialog, NULL, drawAlertDialog,      NULL,
                       NULL,           NULL, uiAlertDialogGetMouse};

AppState ConfirmDialog = {
    idConfirmDialog,      NULL, drawConfirmDialog, NULL, NULL, NULL,
    uiConfirmDialogGetMouse};

AppState EditPage = {
    idEditPage,          initEditPage,       drawEditPage,
    stopEditPage,         uiEditPageGetKeyboard, uiEditPageGetChar,
    uiEditPageGetMouse};

AppState SaveAsPage = {idSaveAsPage,
                      initSaveAsPage,
                      drawSaveAsPage,
                      NULL,
                      uiSaveAsPageGetKeyboard,
                      uiSaveAsPageGetChar,
                      uiSaveAsPageGetMouse};

AppState Explorer = {
    idExplorer,          initExplorer,       drawExplorer,
    stopExplorer,        uiExplorerGetKeyboard, uiExplorerGetChar,
    uiExplorerGetMouse};

AppState HelpPage = {idHelpPage,
                     initHelpPage,
                     drawHelpPage,
                     NULL,
                     uiHelpPageGetKeyboard,
                     uiHelpPageGetChar,
                     uiHelpPageGetMouse};

AppState MainMenu = {idMainMenu,
                     NULL,
                     drawMainMenu,
                     NULL,
                     uiMainMenuGetKeyboard,
                     uiMainMenuGetChar,
                     uiMainMenuGetMouse};

AppState NewPage = {

```

```

    idNewPage,           initNewPage,       drawNewPage,  NULL, uiNewPageGetKeyboard,
    uiNewPageGetChar,   uiNewPageGetMouse};

AppState OpenPage = {idOpenPage,
                     initOpenPage,
                     drawOpenPage,
                     NULL,
                     uiOpenPageGetKeyboard,
                     uiOpenPageGetChar,
                     uiOpenPageGetMouse};

AppState PausePage = {idPausePage, NULL, NULL, NULL, NULL, NULL, NULL};

AppState SimPage = {
    idSimPage,           initSimPage,       drawSimPage,  NULL, uiSimPageGetKeyboard,
    uiSimPageGetChar,   uiSimPageGetMouse};

```

3.4. 函数设计描述

3.4.1. aboutpage (.c, .h)

void drawAboutPage()

功能描述：绘制 AboutPage。

参数描述：没有参数

返回值描述：没有返回值

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void uiAboutPageGetKeyboard(int key, int event)

功能描述：AboutPage 的键盘事件回调函数。

参数描述：键盘事件参数

返回值描述：没有返回值

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiAboutPageGetChar(int ch)`

功能描述: AboutPage 的字符事件回调函数。

参数描述: 字符事件参数

返回值描述: 没有返回值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiAboutPageGetMouse(int x, int y, int button, int event)`

功能描述: AboutPage 的鼠标事件回调函数。

参数描述: 鼠标事件参数

返回值描述: 没有返回值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void gotoAboutPage()`

功能描述: 往状态管理器中压入 AboutPage。

参数描述: 没有参数

返回值描述: 没有返回值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

3.4.2. alertDialog (.c, .h)

```
void setAlertDialog1(char *argv0)
void setAlertDialog2(char *argv0, char *argv1)
void setAlertDialog3(char *argv0, char *argv1, char *argv2)
void setAlertDialog4(char *argv0, char *argv1, char *argv2,
char *argv3)
```

功能描述: 初始化 AlertDialog。

参数描述: 要显示的字符串。

返回值描述: 没有返回值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialog

```
void drawAlertDialog()
```

功能描述: 绘制 AlertDialog。

参数描述: 没有参数

返回值描述: 没有返回值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialog

```
void uiAlertDialogGetMouse(int x, int y, int button, int event)
```

功能描述: AlertDialog 的鼠标事件回调函数。

参数描述: 鼠标事件参数

返回值描述: 没有返回值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dialog

`void gotoAlertDialog()`

功能描述：往状态管理器中压入 AlertDialog。

参数描述：没有参数

返回值描述：没有返回值

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

3.4.3. attempt (.c, .h)

`int isFaceAttempt(int x)`

`int isMoveAttempt(int x)`

`int isUseItemAttempt(int x)`

`int isUseMoveAttempt(int x)`

`int isRemoveItemAttempt(int x)`

`int isRemoveMoveAttempt(int x)`

功能描述：判断指令是否属于某种类型。

参数描述：指令 x

返回值描述：是否属于某种类型

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：位运算

与 3.2 中模块的对应关系：Attempt

`int makeFaceAttempt(Direction dir)`

`int makeMoveAttempt(Direction dir)`

```
int makeUseItemAttempt(int id)
int makeUseMoveAttempt(int id)
int makeRemoveItemAttempt(int id)
int makeRemoveMoveAttempt(int id)
```

功能描述：生成某种类型的指令。

参数描述：指令参数

返回值描述：包含指令参数的对应类型指令

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：位运算

与 3.2 中模块的对应关系：Attempt

```
int argFaceAttempt(int x)
int argMoveAttempt(int x)
int argUseItemAttempt(int x)
int argUseMoveAttempt(int x)
int argRemoveItemAttempt(int x)
int argRemoveMoveAttempt(int x)
```

功能描述：取出某种类型的指令包含的参数。

参数描述：指令 x

返回值描述：指令 x 的参数

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：位运算

与 3.2 中模块的对应关系：Attempt

3.4.4. confirmDialog (.c, .h)

```
void setConfirmDialog1(voidFn value, char *argv0)
void setConfirmDialog2(voidFn value, char *argv0, char *argv1)
```

```
void setConfirmDialog3(voidFn value, char *argv0, char *argv1,  
char *argv2)
```

```
void setConfirmDialog4(voidFn value, char *argv0, char *argv1,  
char *argv2, char *argv3)
```

功能描述: 初始化 ConfirmDialog。

参数描述: 选择确定后要调用的函数 value、要显示的提示内容

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialog

```
void drawConfirmDialog()
```

功能描述: 绘制 ConfirmDialog。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialog

```
void uiConfirmDialogGetMouse(int x, int y, int button, int  
event)
```

功能描述: ConfirmDialog 的鼠标回调函数。

参数描述: 鼠标事件参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialog

void gotoConfirmDialog()

功能描述: 往状态管理器中压入 ConfirmDialog。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialog

3.4.5. controller (.c, .h)

void ScreenRender()

功能描述: 重绘当前状态。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Controller

void bindPlayerMove(void (*_playerMove)(int))

功能描述: 绑定按下各种按键后要触发的函数。

参数描述: 要触发的函数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Controller

`void bindAutoMove(void (*_autoMove)(void))`

功能描述：绑定随时间自动触发的函数。

参数描述：要触发的函数

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Controller

`void playerMoves(int event)`

功能描述：如果没有处于暂停状态，执行 `playerMove`。

参数描述：给 `playerMove` 的参数

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Controller

`void clearTimers()`

功能描述：清除大部分计时器事件。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Controller

`void render(int id)`

功能描述: 根据计时器事件的 ID, 执行相对应的事件函数。

参数描述: 计时器事件的 ID

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Controller

void setPauseBuffer()

功能描述: 设置暂停缓冲时间, 忽略暂停结束后一小段时间的键盘输入。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Controller

void controlKeyboard(int key, int event)

功能描述: 监听键盘事件, 执行相关函数与启动相关计时器。

参数描述: 键盘事件的参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Controller

3.4.6. drawdungeon (.c, .h)

void initDungeonSprites()

功能描述: 读取迷宫相关的图片文件。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Draw

```
static bool inBox(double x, double y, double x1, double x2,  
double y1, double y2)
```

功能描述：判断 (x, y) 是否在由 (x_1, y_1) 与 (x_2, y_2) 构成的矩形中。

参数描述：三个坐标值

返回值描述： (x, y) 是否在由 (x_1, y_1) 与 (x_2, y_2) 构成的矩形中

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：计算几何

与 3.2 中模块的对应关系：Draw

```
void getCellLocation(Dungeon *dungeon, int basex, int basey,  
double size, double lx, double ly, int *_x, int *_y)
```

功能描述：返回屏幕坐标 (lx, ly) 在 $dungeon$ 对应的格子位置。

参数描述：迷宫 $dungeon$ 、镜头位置 $basex$ $basey$ 、迷宫格子尺寸 $size$ 、
屏幕坐标 lx ly 、返回值 $_x$ 、 $_y$

返回值描述：无

重要局部变量定义： x_l x_r y_l y_r

重要局部变量用途描述：根据 lx ly 计算出来的解的粗略上下界

函数算法描述：计算几何

与 3.2 中模块的对应关系：Draw

```
void drawDungeonPokemon(Dungeon *dungeon, int basex, int basey,  
double size, Pokemon *pokemon)
```

功能描述: 在 `dungeon` 对应的格子位置绘制 `pokemon`。

参数描述: 迷宫 `dungeon`、镜头位置 `basex basey`、迷宫格子尺寸 `size`、宝可梦 `pokemon`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 计算几何

与 3.2 中模块的对应关系: Draw

```
void drawDungeon(Dungeon *dungeon, int basex, int basey, double size, int showTag, int showName, DungeonSolution *solution, int enableSolution)
```

功能描述: 绘制 `dungeon`。

参数描述: 迷宫 `dungeon`、镜头位置 `basex basey`、迷宫格子尺寸 `size`、是否展示坐标 `showTag`、是否展示道具和场地事件名 `showName`、迷宫的解 `solution`、是否绘制迷宫的解 `enableSolution`

返回值描述: 无

重要局部变量定义: `visitCount`

重要局部变量用途描述: 绘制迷宫的解时，记录每个格子已经被走了多少次

函数算法描述: 计算几何；链表遍历

与 3.2 中模块的对应关系: Draw

```
void drawDungeonHighlightCellAt(Dungeon *dungeon, int basex, int basey, double size, int x, int y, double length, char *color, double dx, double dy)
```

功能描述: 在 `dungeon` 上指定位置绘制高亮框。

参数描述: 迷宫 `dungeon`、镜头位置 `basex basey`、迷宫格子尺寸 `size`、绘制位置 `x y`、高亮框宽度 `length`、高亮框颜色 `color`、高亮框偏移 `dx dy`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 计算几何

与 3.2 中模块的对应关系: Draw

```
void drawDungeonHighlightCell(Dungeon *dungeon, int basex,  
int basey, double size, double lx, double ly, double length,  
char *color, double dx, double dy)
```

功能描述: 在 dungeon 上屏幕坐标位置绘制高亮框。

参数描述: 迷宫 dungeon、镜头位置 basex basey、迷宫格子尺寸 size、
屏幕坐标位置 lx ly、高亮框宽度 length、高亮框颜色 color、高亮框偏移
dx dy

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 计算几何

与 3.2 中模块的对应关系: Draw

```
int drawDungeonEventEditOverride(LandEvent *landEvent, Item  
*item, double basex, double basey, char *bgcolor, int isEdit,  
int belong, int controlOverride, int *overrideItem, int  
*overrideLandEvent)
```

功能描述: 绘制查看和调整道具与场地事件的栏目。

参数描述: 场地事件 landEvent、道具 item、基准位置 basex basey、背
景颜色 bgcolor、是否能编辑 isEdit、父状态 ID belong、是否调整覆盖
用参数 controlOverride、是否覆盖道具 overrideItem、是否覆盖场地事
件 landEvent

返回值描述: 是否对元素进行了修改

重要局部变量定义: modified

重要局部变量用途描述：是否对元素进行了修改

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Draw

```
int drawDungeonEventEdit(LandEvent *landEvent, Item *item,
double basex, double basey, char *bgcolor, int isEdit, int
belong)
```

功能描述：绘制查看和调整道具与场地事件的栏目，不修改覆盖相关的值。

参数描述：场地事件 landEvent、道具 item、基准位置 basex basey、背景颜色 bgcolor、是否能编辑 isEdit、父状态 ID belong

返回值描述：是否对元素进行了修改

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Draw

```
bool notInAllMenu(double x, double y)
```

功能描述：判断屏幕坐标位置是否在工具栏区。

参数描述：屏幕坐标 x y

返回值描述：屏幕坐标位置是否在工具栏区

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：计算几何

与 3.2 中模块的对应关系：Draw

3.4.7. drawItemBag (.c, .h)

```
int drawItemBag(ItemBag *itemBag, double basex, double basey,
int belong)
```

功能描述：绘制道具背包栏。

参数描述: 道具背包 itemBag、基准位置 baseX baseY、父状态 ID belong

返回值描述: 使用或是丢弃道具的指令

重要局部变量定义: ret、retval

重要局部变量用途描述: 道具编号和道具的使用/丢弃状态

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Draw

3.4.8. dungeon (.c, .h)

`void sortDungeon(Dungeon *dungeon)`

功能描述: 使迷宫合法: 确保 dungeon 大小合法, 确保 dungeon 只有一个开始与结束位置, 清除 dungeon 的开始与结束位置的道具与场地事件, 清除在道具与场地事件正常值外的值。

参数描述: 迷宫 dungeon

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dungeon

`int isDungeonValid(Dungeon *dungeon)`

功能描述: 检查 dungeon 大小是否合法, 是否只有一个开始与结束位置。

参数描述: 迷宫 dungeon

返回值描述: dungeon 是否合法

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dungeon

`int isInDungeon(Dungeon *dungeon, int x, int y)`

功能描述: 检查 (x, y) 是否在 `dungeon` 里。

参数描述: 迷宫 `dungeon`、坐标 x y

返回值描述: (x, y) 是否在 `dungeon` 里

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: `Dungeon`

void getDungeonStart(Dungeon *dungeon, int *_x, int *_y)

功能描述: 获取 `dungeon` 的起点。

参数描述: 迷宫 `dungeon`、起点 x y

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: `Dungeon`

void getDungeonEnd(Dungeon *dungeon, int *_x, int *_y)

功能描述: 获取 `dungeon` 的终点。

参数描述: 迷宫 `dungeon`、终点 x y

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: `Dungeon`

void setDungeonStart(Dungeon *dungeon, int sx, int sy)

功能描述: 设置 `dungeon` 的起点。

参数描述: 迷宫 `dungeon`、起点 sx sy

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dungeon

void setDungeonEnd(Dungeon *dungeon, int sx, int sy)

功能描述：设置 dungeon 的终点。

参数描述：迷宫 dungeon、终点 sx sy

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dungeon

void setDungeonSize(Dungeon *dungeon, int w, int h)

功能描述：调整 dungeon 的大小。会自动调整起点和终点的位置。

参数描述：迷宫 dungeon、宽度 w、高度 h

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dungeon

void setDefaultDungeon(Dungeon *dungeon)

功能描述：把 dungeon 设置成一个默认迷宫。

参数描述：迷宫 dungeon

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dungeon

void randomizeDungeon(Dungeon *dungeon)

功能描述: 把 `dungeon` 设置成一个随机迷宫。

参数描述: 迷宫 `dungeon`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dungeon

void loadDungeon(Dungeon *dungeon, FILE *file)

功能描述: 从 `file` 里读取 `dungeon`。

参数描述: 迷宫 `dungeon`、文件流 `file`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dungeon

void saveDungeon(Dungeon *dungeon, FILE *file)

功能描述: 把 `dungeon` 写入 `file`。

参数描述: 迷宫 `dungeon`, 文件流 `file`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dungeon

3.4.9. dungeonprocess (.c, .h)

`void landEventCalc(Dungeon *dungeon, Pokemon *pokemon)`

功能描述: 处理 pokemon 走到 dungeon 的位置时的场地事件。

参数描述: 迷宫 dungeon、宝可梦 pokemon

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dungeon

`void pokemonStepOn(Dungeon *dungeon, Pokemon *pokemon, ItemBag *itemBag)`

功能描述: 处理有 itemBag 的 pokemon 走到 dungeon 的位置时的捡拾道具事件和场地事件。

参数描述: 迷宫 dungeon、宝可梦 pokemon、道具背包 itemBag

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dungeon

`void pokemonEnemyStepOn(Dungeon *dungeon, Pokemon *pokemon, Item *item)`

功能描述: 处理敌人 pokemon 走到 dungeon 的位置时的捡拾道具事件。敌人不触发场地事件。

参数描述: 迷宫 dungeon、宝可梦 pokemon、道具槽位 item

返回值描述: 无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dungeon

3.4.10. editpage (.c, .h)

`void decEditCellSize()`

功能描述：减少 EditPage 的格子大小。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void incEditCellSize()`

功能描述：增加 EditPage 的格子大小。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void randomizeEditDungeon()`

功能描述：随机化 editDungeon。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void playerMoveEditPage(int event)

功能描述: 根据不同的键盘事件执行不同的流程。

参数描述: 事件编号 event

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 只处理上下左右移动事件，解析为移动镜头位置

与 3.2 中模块的对应关系: Pages

int saveDungeonEditPage()

功能描述: 保存 editDungeon 到文件。

参数描述: 无

返回值描述: 如果成功为 1；如果失败为 -1

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void initEditPage()

功能描述: 初始化 EditPage。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void editGetSolution()`

功能描述: 根据当前的 `editDungeon` 查找解。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void editGetSolutionWithLimit()`

功能描述: 根据当前的 `editDungeon` 查找限定条件下的解。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void drawEditPage()`

功能描述: 绘制 `EditPage`。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void stopEditPage()`

功能描述: 清理 EditPage。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiEditPageGetKeyboard(int key, int event)

功能描述: EditPage 的键盘回调函数。

参数描述: 键盘回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiEditPageGetMouse(int x, int y, int button, int event)

功能描述: EditPage 的鼠标回调函数。

参数描述: 鼠标回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiEditPageGetChar(int ch)

功能描述: EditPage 的输入回调函数。

参数描述: 输入回调参数

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void gotoEditPage()

功能描述：把 EditPage 压入状态管理器中。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void initSaveAsPage()

功能描述：初始化 SaveAsPage。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void drawSaveAsPage()

功能描述：绘制 SaveAsPage。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiSaveAsPageGetKeyboard(int key, int event)

功能描述: SaveAsPage 的键盘回调函数。

参数描述: 键盘回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiSaveAsPageGetChar(int ch)

功能描述: SaveAsPage 的输入回调函数。

参数描述: 输入回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiSaveAsPageGetMouse(int x, int y, int button, int event)

功能描述: SaveAsPage 的鼠标回调函数。

参数描述: 鼠标回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void gotoSaveAsPage()

功能描述: 把 SaveAsPage 压入状态管理器中。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

3.4.11. enemylist (.c, .h)

void clearEnemyList(EnemyList *enemyList)

功能描述: 把 enemyList 设为空的。

参数描述: 敌人列表 enemyList

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 列表操作

与 3.2 中模块的对应关系: Pages

int isEnemyListEmpty(EnemyList *enemyList)

功能描述: 判断 enemyList 是否为空。

参数描述: 敌人列表 enemyList

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 列表操作

与 3.2 中模块的对应关系: Pages

`void emplaceEnemyList(EnemyList *enemyList, Pokemon pokemon)`

功能描述: 在 `enemyList` 里加入 `pokemon`。

参数描述: 敌人列表 `enemyList`、宝可梦 `pokemon`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 列表操作

与 3.2 中模块的对应关系: Pages

`void emplaceEnemyListWithItem(EnemyList *enemyList, Pokemon pokemon, Item item)`

功能描述: 在 `enemyList` 里加入 `pokemon`, 携带道具 `item`。

参数描述: 敌人列表 `enemyList`、宝可梦 `pokemon`、道具 `item`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 列表操作

与 3.2 中模块的对应关系: Pages

`int isOnEnemyList(EnemyList *enemyList, int x, int y)`

功能描述: 查找在 `enemyList` 里有没有人在 `(x, y)` 位置。

参数描述: 敌人列表 `enemyList`、位置 `x y`

返回值描述: 在 `enemyList` 里有没有人在 `(x, y)` 位置

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 列表操作

与 3.2 中模块的对应关系: Pages

`int locationEnemyList(EnemyList *enemyList, int x, int y)`

功能描述: 查找在 enemyList 里在 (x, y) 位置的宝可梦列表位置。

参数描述: 敌人列表 enemyList、位置 x y

返回值描述: 在 enemyList 里在 (x, y) 位置的宝可梦的列表位置

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 列表操作

与 3.2 中模块的对应关系: Pages

void removeAtEnemyList(EnemyList *enemyList, size_t id)

功能描述: 移除 enemyList 里的第 id 个宝可梦。

参数描述: 敌人列表 enemyList、位置 id

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 列表操作

与 3.2 中模块的对应关系: Pages

void setItemAtEnemyList(EnemyList *enemyList, size_t id, Item item)

功能描述: 把 enemyList 里的第 id 个宝可梦的持有道具变成 item。

参数描述: 敌人列表 enemyList、位置 id、道具 item

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 列表操作

与 3.2 中模块的对应关系: Pages

3.4.12. explorer (.c, .h)

void checkManaphyHealth()

功能描述: 检查玛纳霏的状态。如果他倒下了，那么游戏就会结束。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`int manhattanDistance(int x1, int y1, int x2, int y2)`

功能描述: 计算 (x_1, y_1) 和 (x_2, y_2) 之间的曼哈顿距离。

参数描述: 坐标 x_1 y_1 x_2 y_2

返回值描述: (x_1, y_1) 和 (x_2, y_2) 之间的曼哈顿距离

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 计算几何

与 3.2 中模块的对应关系: Pages

`int enemyCanStepIn(int dx, int dy)`

功能描述: 判断一个敌人是否可以进入迷宫的 (dx, dy) 。

参数描述: 坐标 dx dy

返回值描述: 敌人是否可以进入迷宫的 (dx, dy)

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`int spawnSingleEnemy(int distance)`

功能描述: 尝试生成一个敌人，他/她离玛纳霏的曼哈顿距离要大于 $distance$ 。

参数描述: 距离阈值 $distance$

返回值描述: 敌人是否生成成功

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void enemyRound()

功能描述: 进行敌人的回合。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void decreaseManaphyBelly(double val)

功能描述: 减少玛纳霏的饱腹度 `val` 点，并执行连带的流程。

参数描述: 减少的饱腹度 `val`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

int manaphyMove(int att)

功能描述: 让玛纳霏执行指令 `att`。

参数描述: 指令 `att`

返回值描述: 这个指令是否消耗了他的回合

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void manaphyRound(int att)

功能描述: 进行玛纳霏的回合，让他执行指令 att。

参数描述: 指令 att

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void manaphyMoveAttempt(int event)

功能描述: 根据事件编码指令，尝试进行玛纳霏的回合。

参数描述: 事件 event

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 接收的事件有改变朝向、移动和使用招式

与 3.2 中模块的对应关系: Pages

void giveCheat()

功能描述: 提示下一步往哪边走更优。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 调用求解模块检查当前周围的四个格子哪个更优。如果有钥匙被

偷的情况，就将目标从终点改为持有钥匙的敌人。

与 3.2 中模块的对应关系：Pages

void spawnEnemy()

功能描述：尝试一次性生成 `spawnEnemyCount` 个敌人，不能刷到玛纳霏贴脸的位置。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void initExplorer()

功能描述：初始化 Explorer。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void drawExplorer()

功能描述：绘制 Explorer。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系: Pages

`void stopExplorer()`

功能描述: 清理 Explorer。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiExplorerGetKeyboard(int key, int event)`

功能描述: Explorer 的键盘回调函数。

参数描述: 键盘回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiExplorerGetChar(int ch)`

功能描述: Explorer 的输入回调函数。

参数描述: 输入回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiExplorerGetMouse(int x, int y, int button, int event)`

功能描述: Explorer 的鼠标回调函数。

参数描述: 鼠标回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void gotoExplorer()

功能描述: 把 Explorer 压入状态管理器中。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

3.4.13. **getfilename dialog (.c, .h)**

void initGetFileNameDialog(char *name)

功能描述: 初始化 GetFileNameDialog。

参数描述: 默认文件名 name

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialogs

int drawGetFileNameDialog()

功能描述: 绘制 GetFileNameDialog。

参数描述：无

返回值描述：文本是否被编辑了

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dialogs

3.4.14. **helpList (.c, .h)**

void clearHelpList()

功能描述：将 helpList 设为空的。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Draws

void addHelpEntry(char *fun, char *key)

功能描述：在 helpList 里加一条 (fun, key) 的字符串对。

参数描述：要显示的字符串 fun key

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Draws

void drawHelpList(double basex, double topy)

功能描述：绘制 helpList。

参数描述：基准位置 basex basey

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Draws

3.4.15. helpPage (.c, .h)

`void readHelpPageSprites()`

功能描述：读取 HelpPage 所需要的图像资源。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Draws

`void initHelpPage()`

功能描述：初始化 HelpPage。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Draws

`void drawHelpPage()`

功能描述：绘制 HelpPage。

参数描述：无

返回值描述：无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Draws

void uiHelpPageGetKeyboard(int key, int event)

功能描述: HelpPage 的键盘回调函数。

参数描述: 键盘回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Draws

void uiHelpPageGetChar(int ch)

功能描述: HelpPage 的输入回调函数。

参数描述: 输入回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Draws

void uiHelpPageGetMouse(int x, int y, int button, int event)

功能描述: HelpPage 的鼠标回调函数。

参数描述: 鼠标回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Draws

void gotoHelpPage()

功能描述: 把 HelpPage 压入状态管理器。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Draws

3.4.16. imagesupport (.c, .h)

HBITMAP readBmpImage(char *fileName)

功能描述: 从 fileName 读取一个 bmp 文件，存储为 HBITMAP 对象。

参数描述: 文件名 filename

返回值描述: 读取的 bmp 文件

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Graphics

3.4.17. itembag (.c, .h)

void clearItemBag(ItemBag *itemBag)

功能描述: 把 itemBag 设为空的。

参数描述: 道具背包 itemBag

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`int addIntoItemBag(ItemBag *itemBag, Item item)`

功能描述：往 `itemBag` 里面加入 `item`。

参数描述：道具背包 `itemBag`、道具 `item`

返回值描述：是否成功加入

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`int removeOutItemBag(ItemBag *itemBag, size_t index)`

功能描述：把 `itemBag` 里面第 `index` 个道具移除。

参数描述：道具背包 `itemBag`、位置 `index`

返回值描述：是否成功移除

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`int getKeyInItemBag(ItemBag *itemBag)`

功能描述：获取 `itemBag` 里面的钥匙状态。

参数描述：道具背包 `itemBag`

返回值描述：`itemBag` 里面的钥匙状态

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：位运算

与 3.2 中模块的对应关系：Pages

`void sortItemBag(ItemBag *itemBag)`

功能描述: 把 itemBag 里面的道具排序。

参数描述: 道具背包 itemBag

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 快速排序

与 3.2 中模块的对应关系: Pages

3.4.18. items (.c, .h)

`void readItemSprites(int num)`

功能描述: 读取编号为 num 的道具的图像。

参数描述: 编号 num

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

`void initItems()`

功能描述: 初始化道具数据。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

`int cmpItems(const void *lhs, const void *rhs)`

功能描述: 比较两个道具的先后顺序。

参数描述: 道具 `lhs rhs`

返回值描述: 如果 `lhs` 小于 `rhs`: 返回小于零的值; 如果 `lhs` 等于 `rhs`: 返回零; 如果 `lhs` 大于 `rhs`: 返回大于零的值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

3.4.19. landevent (.c, .h)

`void initLandEvents()`

功能描述: 初始化场地事件数据。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

3.4.20. main (.c)

`void Main()`

功能描述: 主函数。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Main

3.4.21. messagedialog (.c, .h)

`void setMessage(char *s)`

功能描述：设置消息字符串为 s。

参数描述：消息 s

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dialog

`void emplaceMessage(char *s)`

功能描述：在消息字符串末尾增加 s。

参数描述：消息 s

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dialog

`int isMessageEmpty()`

功能描述：判断消息字符串是否是空的。

参数描述：无

返回值描述：消息字符串是否是空的

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Dialog

`void clearMessage()`

功能描述: 清空消息字符串。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialog

void drawMessageDialog()

功能描述: 绘制 MessageDialog。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Dialog

3.4.22. pages (.c, .h)

void callingExitWarning(voidFn nex)

功能描述: 执行函数 nex。在这之前，如果用户没有保存文件，那么警告用户；待用户确认后执行 nex。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void smPopStateUntilMainMenu()

功能描述: 弹出状态管理器的状态，直到栈顶是 `MainMenu`。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: `Pages`

void drawToolsBar()

功能描述: 绘制工具栏。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 根据当前状态管理器的栈顶绘制不同的选项

与 3.2 中模块的对应关系: `Pages`

void drawMainMenu()

功能描述: 绘制 `MainMenu`。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: `Pages`

void initOpenPage()

功能描述: 初始化 `OpenPage`。

参数描述: 无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void drawOpenPage()

功能描述：绘制 OpenPage。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void initNewPage()

功能描述：初始化 NewPage。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void drawNewPage()

功能描述：绘制 NewPage。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiMainMenuGetKeyboard(int key, int event)

功能描述: MainMenu 的键盘回调函数。

参数描述: 键盘回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiMainMenuGetChar(int ch)

功能描述: MainMenu 的输入回调函数。

参数描述: 输入回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiMainMenuGetMouse(int x, int y, int button, int event)

功能描述: MainMenu 的鼠标回调函数。

参数描述: 鼠标回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiNewPageGetKeyboard(int key, int event)`

功能描述: NewPage 的键盘回调函数。

参数描述: 键盘回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiNewPageGetChar(int ch)`

功能描述: NewPage 的输入回调函数。

参数描述: 输入回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiNewPageGetMouse(int x, int y, int button, int event)`

功能描述: NewPage 的鼠标回调函数。

参数描述: 鼠标回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

`void uiOpenPageGetKeyboard(int key, int event)`

功能描述: OpenPage 的键盘回调函数。

参数描述: 键盘回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiOpenPageGetChar(int ch)

功能描述: OpenPage 的输入回调函数。

参数描述: 输入回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiOpenPageGetMouse(int x, int y, int button, int event)

功能描述: OpenPage 的鼠标回调函数。

参数描述: 鼠标回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void gotoNewPage()

功能描述: 往状态管理器中压入 NewPage。

参数描述: 无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

void gotoOpenPage()

功能描述：往状态管理器中压入 OpenPage。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

3.4.23. pausepage (.c, .h)

int isPausing()

功能描述：判断现在是否处于暂停状态。

参数描述：无

返回值描述：现在是否处于暂停状态

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Controller

void makePause(double seconds)

功能描述：创建一个时长为 seconds 秒的暂停状态。

参数描述：时长 seconds

返回值描述：无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 劫持控制流，在循环内不断处理绘制函数并刷新，在期间接收键盘输入信号但是将其全部弃置。

与 3.2 中模块的对应关系: Controller

3.4.24. pokemon (.c, .h)

void readPokemonSprites(int num)

功能描述: 读取图鉴编号为 num 的宝可梦的头像和迷宫内形象。

参数描述: 图鉴编号 num

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

void initPokedex()

功能描述: 初始化宝可梦图鉴数据。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

void initMovedex()

功能描述: 初始化招式图鉴数据。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

double getBaseStat(int lv, double base, double growth)

功能描述: 计算等级为 `lv`、基础值为 `base`、成长为 `growth` 时的基础能力值。

参数描述: 等级 `lv`、基础值 `base`、成长 `growth`

返回值描述: 基础能力值

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

double calcDamage(int lv, double effect, double atk, double def)

功能描述: 计算攻击方等级为 `lv`、招式效果为 `effect`、攻击方攻击为 `atk`、防御方防御为 `def` 时的基础伤害。

参数描述: 等级 `lv`、招式效果 `effect`、攻击方攻击 `atk`、防御方防御 `def`

返回值描述: 基础伤害

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Database

double calcExp(int aLv, int bLv)

功能描述: 计算等级为 `aLv` 的宝可梦打败了等级为 `bLv` 的宝可梦时获得的经验值。

参数描述: 等级 `aLv` `bLv`

返回值描述：获得经验值

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Database

`void makePokemonStatBound(Pokemon *pokemon)`

功能描述：让 `pokemon` 的参数处于合法区间。

参数描述：宝可梦 `pokemon`

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Database

`int updatePokemonStat(Pokemon *pokemon)`

功能描述：计算 `pokemon` 是否升级，更新其能力值。

参数描述：宝可梦 `pokemon`

返回值描述：等级是否提升了

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Database

`void spawnPokemon(Pokemon *pokemon, Role role, int species, int gender)`

功能描述：把 `pokemon` 设为等级为 1、阵营为 `role`、种族为 `species`、性别为 `gender` 的宝可梦。

参数描述：宝可梦 `pokemon`、阵营 `role`、种族 `species`、性别 `gender`

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Database

3.4.25. simpage (.c, .h)

`void decSimCellSize()`

功能描述：减少 SimPage 的格子大小。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void incSimCellSize()`

功能描述：增加 SimPage 的格子大小。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void checkCresseliaHealth()`

功能描述：更新克雷色利亚的状态。如果她倒下了，那么停止自动执行。

参数描述：无

返回值描述：无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void playerMoveSimPage(int event)

功能描述: 处理事件 event 对应的流程。

参数描述: 事件 event

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 只关心移动的事件，视为移动镜头。

与 3.2 中模块的对应关系: Pages

int cresseliaAttempt()

功能描述: 让克雷色利亚决定接下来的指令。

参数描述: 无

返回值描述: 克雷色利亚接下来的指令

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void cresseliaMove()

功能描述: 进行克雷色利亚的回合。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void initSimPage()`

功能描述：初始化 SimPage。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void startAutoSimulating()`

功能描述：开始自动执行。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void endAutoSimulating()`

功能描述：结束自动执行。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void setSimulateSpeed(int speed)`

功能描述：把自动执行的速度倍率设为 `speed`。

参数描述：速度倍率 `speed`

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void drawSimPage()`

功能描述：绘制 `SimPage`。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void uiSimPageGetMouse(int x, int y, int button, int event)`

功能描述： `SimPage` 的鼠标回调函数。

参数描述： 鼠标回调参数

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：Pages

`void uiSimPageGetKeyboard(int key, int event)`

功能描述： `SimPage` 的键盘回调函数。

参数描述: 键盘回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void uiSimPageGetChar(int ch)

功能描述: SimPage 的输入回调函数。

参数描述: 输入回调参数

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

void gotoSimPage()

功能描述: 往状态管理器里压入 SimPage。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Pages

3.4.26. solveModel (.c, .h)

void clearHeap()

功能描述: 将二叉堆设为空的。

参数描述: 无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：堆操作

与 3.2 中模块的对应关系：Algorithm

int isEmpty()

功能描述：判断二叉堆是否为空。

参数描述：无

返回值描述：二叉堆是否为空

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：堆操作

与 3.2 中模块的对应关系：Algorithm

void pushUpHeap(int x)

功能描述：向上更新编号为 x 的二叉堆结点。

参数描述：结点编号 x

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：堆操作

与 3.2 中模块的对应关系：Algorithm

void pushDownHeap(int x)

功能描述：向下更新编号为 x 的二叉堆结点。

参数描述：结点编号 x

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述: 无

函数算法描述: 堆操作

与 3.2 中模块的对应关系: Algorithm

void popHeap()

功能描述: 弹出二叉堆的堆顶。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 堆操作

与 3.2 中模块的对应关系: Algorithm

void emplaceHeap(HeapNode node)

功能描述: 往二叉堆里面加入结点 node。

参数描述: 结点 node

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 堆操作

与 3.2 中模块的对应关系: Algorithm

HeapNode topHeap()

功能描述: 返回二叉堆的根结点。

参数描述: 无

返回值描述: 二叉堆的根结点

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 堆操作

与 3.2 中模块的对应关系: Algorithm

`RouteNode *newRouteNode(int x, int y)`

功能描述: 生成一个路线链表的结点, 内容为 (x, y)。

参数描述: 坐标 x y

返回值描述: 新的链表节点

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Algorithm

`int xy2a(int x, int y, int h, int hasKey, int allKey)`

功能描述: 将位置 (x, y) 和钥匙状态编码。

参数描述: 坐标 x y、迷宫高度 h、钥匙状态 hasKey、钥匙总数 allKey

返回值描述: 编码结果

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Algorithm

`void a2xy(int a, int h, int allKey, int *x, int *y, int *hasKey)`

功能描述: 将位置 (x, y) 和钥匙状态解码。

参数描述: 编码 a、迷宫高度 h、钥匙总数 allKey、坐标 x y、钥匙状态 hasKey

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Algorithm

```
void clearDungeonSolution(DungeonSolution *solution)
```

功能描述: 清除 `solution`。

参数描述: 迷宫解法 `solution`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: `Algorithm`

```
void makeDijkstra(Dungeon *dungeon, int sid, int tid, int allKey, lint hpPenalty, lint limit)
```

功能描述: 在钥匙总数为 `allKey`、HP 损失惩罚为 `hpPenalty`、限制距离为 `limit` 的情况下，在 `dungeon` 上进行迪科斯彻算法。

参数描述: 迷宫 `dungeon`、开始结点编号 `sid`、目标结点编号 `tid`、钥匙总数 `allKey`、HP 损失惩罚 `hpPenalty`、限制距离 `limit`

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 进行迪科斯彻算法，具体描述见前文

与 3.2 中模块的对应关系: `Algorithm`

```
int getDungeonSolutionWithLimit(Dungeon *dungeon, DungeonSolution *solution, lint limit, lint hpPenalty)
```

功能描述: 在钥匙总数为 `allKey`、HP 损失惩罚为 `hpPenalty`、限制距离为 `limit` 的情况下，找出 `dungeon` 的解法，存储到 `solution`。

参数描述: 迷宫 `dungeon`、迷宫解法 `solution`、限制距离 `limit`、HP 损失惩罚 `hpPenalty`

返回值描述: 是否有从起点到终点的路径

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 利用迪科斯彻算法计算

与 3.2 中模块的对应关系: Algorithm

```
int getDungeonSolution(Dungeon *dungeon, DungeonSolution  
*solution)
```

功能描述: 在默认 HP 损失惩罚、不限制距离的情况下，找出 dungeon 的解法，存储到 solution。

参数描述: 迷宫 dungeon、迷宫解法 solution、限制距离 limit、HP 损失惩罚 hpPenalty

返回值描述: 是否有从起点到终点的路径

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 利用迪科斯彻算法计算；起点到终点的路径为从终点到起点反向建立链表

与 3.2 中模块的对应关系: Algorithm

```
lint getDungeonDistance(Dungeon *dungeon, int sx, int sy, int  
skey, int tx, int ty, lint hpPenalty, int enableKey)
```

功能描述: 在 HP 损失惩罚为 hpPenalty、不限制距离、拥有钥匙状态为 skey、使用钥匙的情况为 enableKey 的情况下，找出从 (sx, sy) 到 (tx, ty) 的最短距离。

参数描述: 迷宫 dungeon、起点 sx sy、拥有钥匙状态 skey、终点 tx ty、HP 损失惩罚 hpPenalty、使用钥匙的情况 enableKey

返回值描述: 从 (sx, sy) 到 (tx, ty) 的最短距离

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 利用迪科斯彻算法计算

与 3.2 中模块的对应关系: Algorithm

3.4.27. statemanager (.c, .h)

`void smInit()`

功能描述： 初始化状态管理器。

参数描述： 无

返回值描述： 无

重要局部变量定义： 无

重要局部变量用途描述： 无

函数算法描述： 栈操作

与 3.2 中模块的对应关系： StateManager

`bool smIsEmpty()`

功能描述： 判断状态管理器是否是空的。

参数描述： 无

返回值描述： 状态管理器是否是空的

重要局部变量定义： 无

重要局部变量用途描述： 无

函数算法描述： 栈操作

与 3.2 中模块的对应关系： StateManager

`int smStateCount()`

功能描述： 获取当前状态管理器里面的状态数量。

参数描述： 无

返回值描述： 状态管理器里面的状态数量

重要局部变量定义： 无

重要局部变量用途描述： 无

函数算法描述： 栈操作

与 3.2 中模块的对应关系： StateManager

`AppState *smStateTop()`

功能描述: 获取当前状态管理器的栈顶状态。

参数描述: 无

返回值描述: 状态管理器的栈顶状态

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 栈操作

与 3.2 中模块的对应关系: StateManager

void callCtor(AppState *now)

功能描述: 调用状态的构建函数。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: StateManager

void callProc(AppState *now)

功能描述: 调用状态的运行函数。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: StateManager

void callDtor(AppState *now)

功能描述: 调用状态的析构函数。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: StateManager

void smLastProc()

功能描述: 调用状态管理器栈顶的后一个状态的构建函数。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 栈操作

与 3.2 中模块的对应关系: StateManager

void smTopChanging()

功能描述: 解除当前状态的构建函数与计时器回调函数的绑定。

参数描述: 无

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: StateManager

void smRebuildTop()

功能描述: 调用栈顶状态的构建函数，注册栈顶状态的三个回调函数，把栈顶状态的运行函数与计时器回调函数绑定。

参数描述: 无

返回值描述: 无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：仅业务代码

与 3.2 中模块的对应关系：StateManager

void smPushState(AppState *state)

功能描述：把 state 压入状态管理器的栈顶。进行相应函数的绑定和解绑。

参数描述：状态 state

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：栈操作

与 3.2 中模块的对应关系：StateManager

void smBarePushState(AppState *state)

功能描述：只把 state 压入状态管理器的栈顶。

参数描述：状态 state

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：栈操作

与 3.2 中模块的对应关系：StateManager

void smPopState()

功能描述：弹出状态管理器的栈顶。进行相应函数的绑定和解绑。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：栈操作

与 3.2 中模块的对应关系：StateManager

void smBarePopState()

功能描述：只弹出状态管理器的栈顶。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：栈操作

与 3.2 中模块的对应关系：StateManager

void smPopStateUntil(int target)

功能描述：一直弹出状态管理器的栈顶，直到栈顶状态编号为 target。

参数描述：状态编号 target

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：栈操作

与 3.2 中模块的对应关系：StateManager

void smClearState()

功能描述：把状态管理器的栈弹空。

参数描述：无

返回值描述：无

重要局部变量定义：无

重要局部变量用途描述：无

函数算法描述：栈操作

与 3.2 中模块的对应关系：StateManager

3.4.28. statusbar (.c, .h)

```
void drawStatusBar(Pokemon *pokemon, double basex, double basey, int portrait, int belong)
```

功能描述: 绘制 pokemon 的状态栏。可选是否绘制头像。

参数描述: 宝可梦 pokemon、基础位置 basex basey、绘制头像 portrait、父状态 ID belong

返回值描述: 无

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Draw

```
int drawMoveList(Pokemon *pokemon, double basex, double basey, int belong)
```

功能描述: 绘制 pokemon 的招式栏。

参数描述: 宝可梦 pokemon、基础位置 basex basey、父状态 ID belong

返回值描述: 使用或忘记招式的指令

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Draw

3.4.29. utils (.c, .h)

```
int pick(int a, int b)
```

功能描述: 返回 a 的第 b 个二进制位。

参数描述: a b

返回值描述: a 的第 b 个二进制位

重要局部变量定义: 无

重要局部变量用途描述: 无

函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Utils

int shl(int a)

功能描述: 返回 1 左移 a 位。

参数描述: a

返回值描述: 1 左移 a 位

重要局部变量定义: 无

重要局部变量用途描述: 无

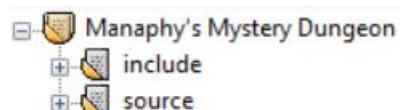
函数算法描述: 仅业务代码

与 3.2 中模块的对应关系: Utils

3.5. 源代码文件组织设计

3.5.1. 文件目录结构

本程序在 Dev-C++ 内的文件目录如图。





3.5.2. 文件函数结构

.c 文件与 .h 文件的内容已在之前的章节尽数展示，在此不作重复。

3.5.3. 多文件构成机制

本项目的文件之间采用 `#include` 文件所对应的 `.h` 文件来连接各个 `.c` 文件。`extern` 声明绝大部分都放在 `.h` 文件中，而真实存放位置是其对应的 `.c` 文件；引用 `.h` 文件就可以访问到对应的变量。

本项目的大部分 `.h` 文件采用了 `#pragma once` 来确保它只会被包含一次；其它文件采用了 `#define` 保护的方法，虽然两种方法的效果是一致的。

4. 部署运行和使用说明

4.1. 编译安装

4.1.1. 使用 Dev C++

请解压特供的 `MMD_devcpp.zip` 文件，使用 Dev C++ 打开 `Manaphy's Mystery Dungeon.dev`。

确认编译时采用了 GNU C99 标准。建议调整优化级别到 Highest (`-Ofast`)。



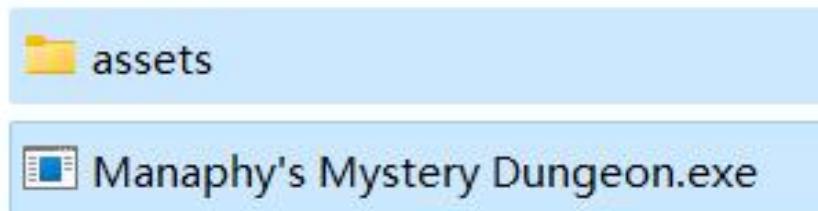
务必在链接器中加入 `-lgdi32` 和 `-lopengl32`，链接必要的库。

在连接器命令行加入以下命令

-static-libgcc -lgdi32 -lopengl32

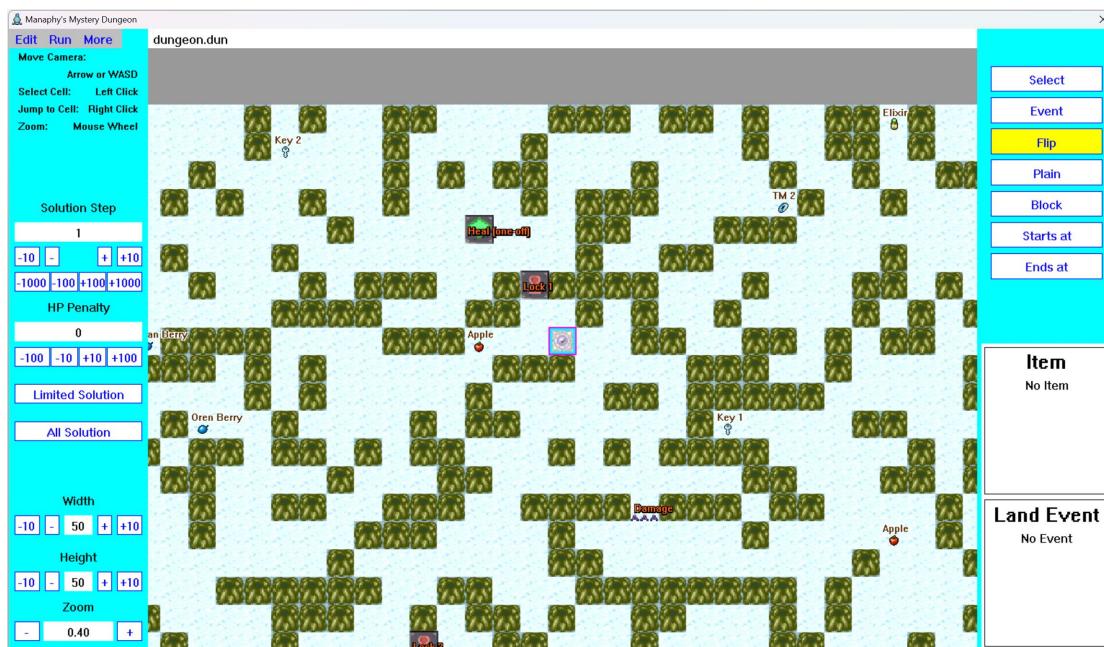
编译后即可获得 Manaphy's Mystery Dungeon.exe。

确保可执行文件和 assets 文件夹处于同一级别。

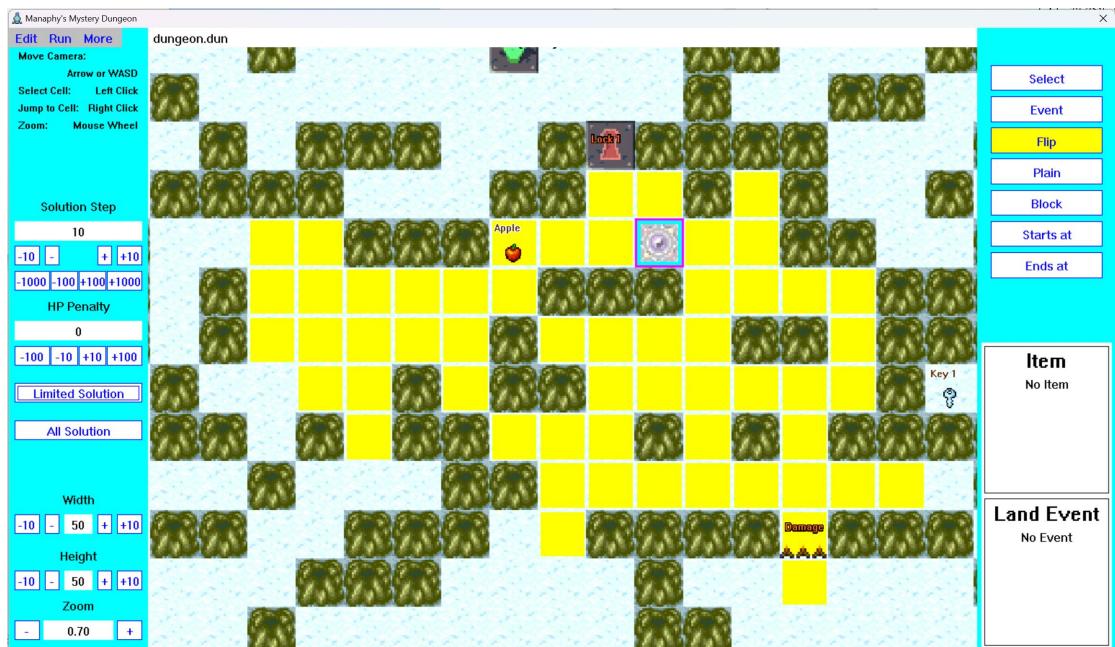


4.2. 运行测试

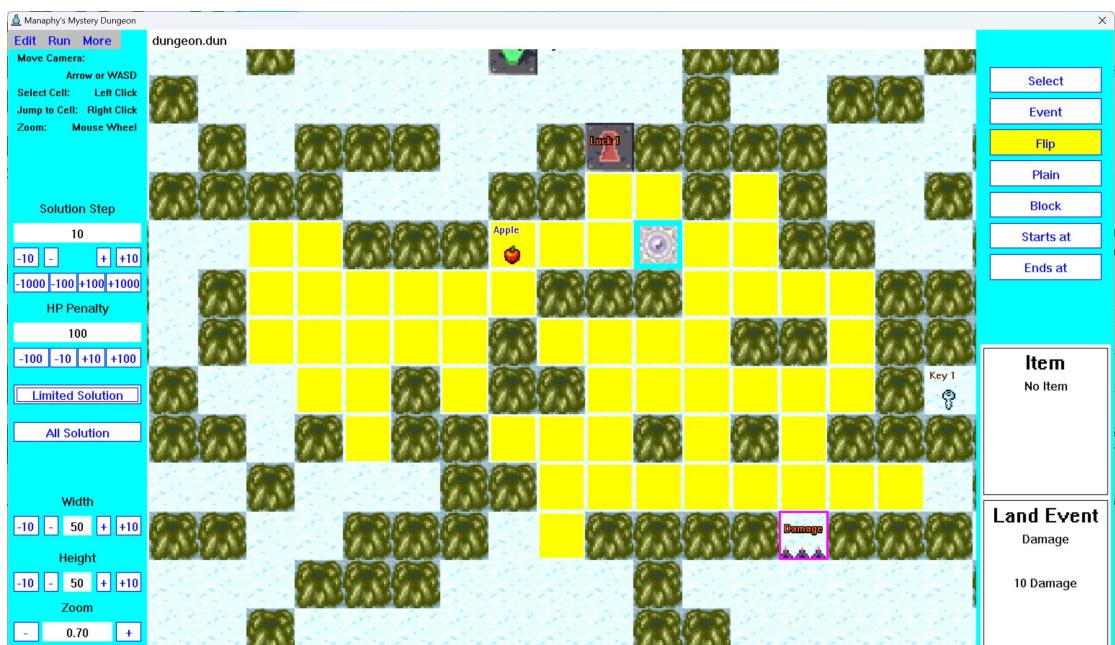
打开样例迷宫 dungeon.dun。



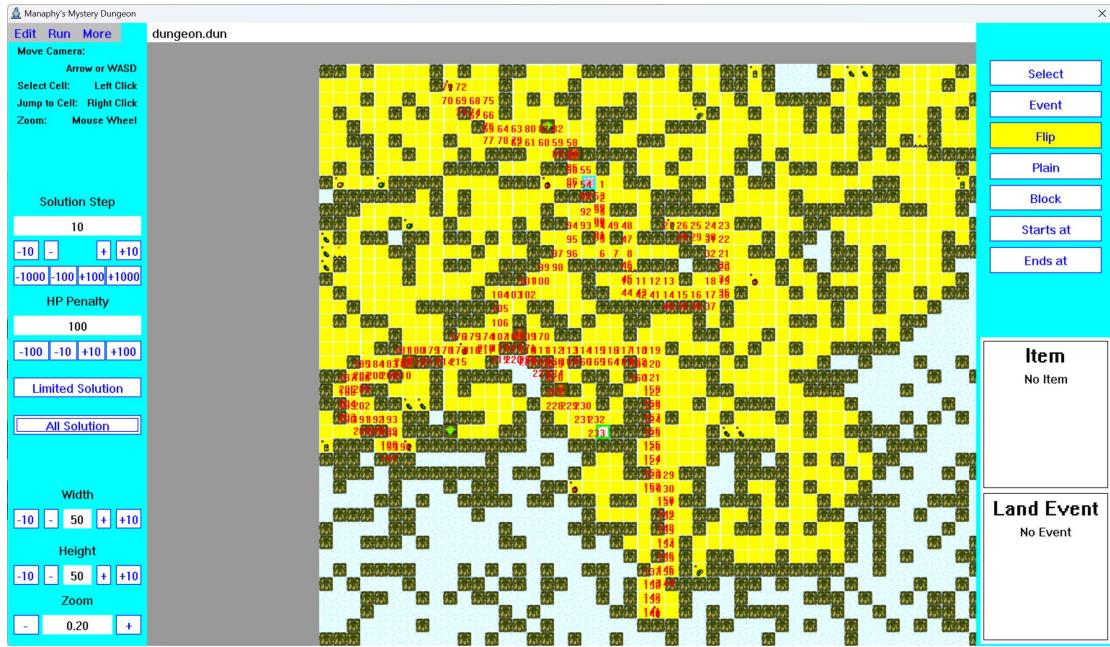
设置 Solution Step 为 10, 点击 Limited Solution 查看可以到达的位置。



调整 HP Penalty 到 100, 点击 Limited Solution , 发现它这次没有选择走右下角的伤害格子。



点击 All Solution 查看迷宫的解。



选择工具栏的 Run > Auto Run, 进入自动执行页面。点击 Lock Camera 后点击 Auto Run, 观看克雷色利亚探索迷宫的过程。

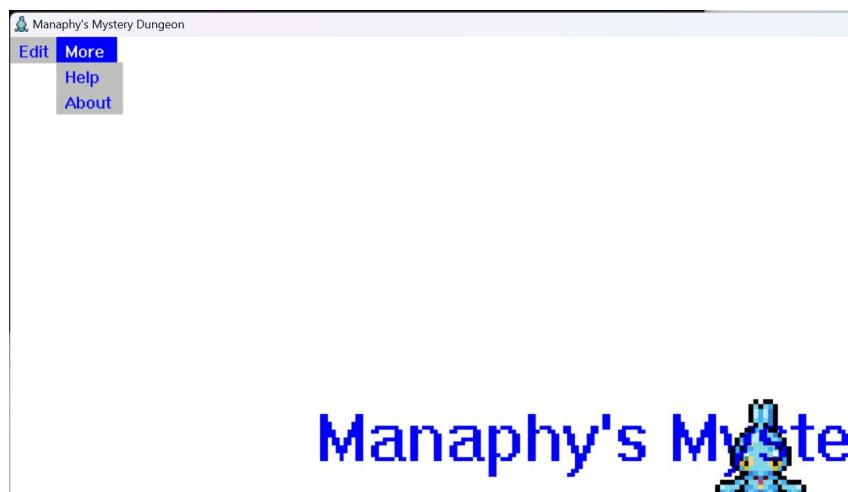


选择工具栏的 Run > Explore, 进入游玩页面。打开 Auto Spawn, 在有敌人生的情况下进行游玩。注意右侧的铁炮鱼。



4.3. 用户使用手册

如果你是第一次使用这个程序，请先选择左上角工具栏的 More > Help 查看帮助。帮助内有人物、迷宫、场地事件和道具的介绍。



你可以先从样例迷宫开始。选择 Edit > New 或者 Edit > Open，打开已存在的样例迷宫 `dungeon.dun`。



进入编辑页面。页面的按键指南在左上角常驻，你可以随时查阅。



中间为迷宫视图；左上角为操作指引；左侧中部为求解模块的前端；左下角为改变迷宫大小与改变视图缩放；右上角为编辑模式；右下角为迷宫的道具和场地事件编辑。

通过工具栏或快捷键可以进行保存等操作。如果当前迷宫还没有保存，顶上的显示迷宫文件名的条会变黄。

求解模块前端可以决定输出在具体第几步时可以到达的位置（你可以通过这个来查看求解中间过程的可视化），也可以求出所有可以到达的位置；另外可以改变 HP 惩罚（HP Penalty）的值，这个值越大，程序会越避免走那些会导致 HP 损失的格子。求解会自动保存迷宫。

点按改变迷宫大小部分的相应按键可以改变迷宫的长度和宽度。

点按改变视图缩放的相应按键可以改变视图的缩放。另外也可以使用鼠标滚轮调整。

点按编辑模式可以改变当前的编辑模式。编辑模式有 7 种：

1. 选择 (Select)：选择一个格子，改变其道具和场地事件。
2. 事件 (Event)：用指定的道具和/或场地事件覆盖指定格子的相应值。
3. 翻转 (Flip)：让格子在平地和障碍之间改变。
4. 平地 (Plain)：把选定的格子从障碍设为平地。
5. 障碍 (Block)：把选定的格子从平地设为障碍。
6. 起点 (Starts at)：设置起点的位置。
7. 终点 (Ends at)：设置终点的位置。

在选择模式下，点击右下角的道具和场地事件编辑栏会改变当前选择的格子的相关值。



在事件模式下，点击右下角的道具和场地事件编辑栏会改变是否要覆盖原有的值，以及要覆盖的相关值。



在其他模式下，编辑栏会显示当前鼠标所在的格子的道具和场地事件，不能修改。



在工具栏的 Edit 中有更多的文件操作。其中，Randomize 会以当前迷宫的大小随机生成一个迷宫，并覆盖当前迷宫。

选择 Run > Auto Run, 进入自动执行页面。



在自动执行页面中，会请克雷色利亚来让她尝试自动完成迷宫。

中间为迷宫视图；左上角为操作指引；左侧中上部为提升克雷色利亚等级和恢复克雷色利亚的选项；中部为自动演示相关的选项；左下角为克雷色利亚的状态栏；右上角为她的道具背包；右下角为她的招式列表。

迷宫视图会记录克雷色利亚走过的路。

如果提升克雷色利亚的等级，会立即刷新她的能力；如果恢复克雷色利亚，将回复她的所有 HP 和饱腹度。但如果克雷色利亚已经倒下，这些都不会起效。

左侧中部可以调整自动演示的速度，以及开/关自动演示、进行单步演示，以及是否让视角锁定到克雷色利亚上。

注意克雷色利亚不会使用除了钥匙之外的道具；因为没有敌人，克雷色利亚也不会使用招式。克雷色利亚会尽量避免损失 HP。

在自动执行页面执行中途呼出其他页面（比如帮助页面和关于页面）会打断目前正在执行的自动演示。

自动执行页面会自动保存迷宫到文件。

选择 Run > Explore，进入游玩页面。



在游玩页面中，你将以玛纳霏的身份完成这个迷宫。注意在这个期间视角和缩放都是固定的。

中间为迷宫视图；左上角为操作指引；左侧中部为生成敌人相关的选项；左下角为玛纳霏的状态栏；右上角为道具背包；右下角为招式列表。另外在工具栏中有作弊选项。

你需要进行移动、使用道具和使用招式来攻击敌人。

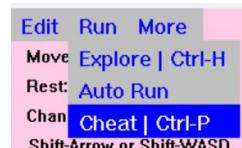
如果点击生成敌人，那么会立即在迷宫中尝试生成指定数量的敌人；如果打开自动生成敌人，每回合结束时会有一定概率在视野范围外生成敌人。生成敌人的强度和玛纳霏当前等级有关。

玛纳霏会自己捡拾道具。在道具背包点击相关道具可以使用它们。注意敌人也会捡拾道具，不过他们只能同时持有一个道具。持有钥匙的敌人将会被高亮。

在攻击敌人前，请先正对敌人，然后对他们使用招式。打倒敌人会获得他们所持有的道具。

作弊选项可以提示下一步应该走的方向。如果有钥匙被偷的情况，作弊选项也会有相应的提示。

游玩页面会自动保存迷宫到文件。



5. 团队合作

5.1. 开发日志

第一天：确定选题，搜集资料，学习探索 `LibGraphics` 的使用。

第二天：请来玛纳霏。确定项目框架。编写状态管理器和编辑页面。

第三天：初步完成编辑页面。

第四天：编写游玩页面和自动执行页面。编写相关算法。

第五天：请来克雷色利亚。初步完成游玩页面和自动执行页面。

第六天：编写道具、场地事件和招式相关内容。

第七天：编写道具、场地事件和招式相关逻辑；完善页面内容。

第八天：编写敌人相关逻辑。开始给 UI 贴图。

第九天：完成 UI 翻新，进一步完善页面内容。项目宣告完成。

第十天：写实验报告。

5.2. 编码规范

统一采用 `clang-format` 格式化代码，采用的标准为 LLVM。

1. 所有的大括号都不换行。
2. 使用空格缩进，缩进宽度为 2。
3. 变量和函数的命名采用小驼峰命名法；结构和定义采用大驼峰命名法。
4. 声明指针时的星号 “*” 靠近变量名，而距离类型名有 1 个空格。
5. 表达式中不允许省略任何括号。
6. 宏定义如果可以用括号包含的，要用括号包含。
7. 对于从程序外部输入的数据，必须进行合法性检查并做相应处理。

5.3. 任务分工

****:

1. 游戏部分的数值设计
2. 编辑器框架的搭建

3. 编辑器函数的编写
4. 界面的设计
5. 编辑器所使用的素材的收集与处理

5.4. 个人遇到的难点与解决方案

难点：暂停功能阻塞键盘输入信号，导致定时器事件错误启停与指令混乱。

解决方案：被克雷色利亚托梦，优化了暂停相关逻辑：把单纯的死循环改写为劫持控制流，在循环内不断处理绘制函数并刷新，在期间接收键盘输入信号但是将其全部弃置。这样比较优秀地解决了问题。

5.5. 合作总结

5.5.1. 开发亮点

全程使用 `LibGraphics` 实现了一个附带有自动求解和内置运行器的迷宫编辑器。

5.5.2. 开发挑战点

1. 所需要实现的内容极多，代码量大，对编程能力要求高。
2. 前一点带来的多文件工程管理需要对程序架构有深刻了解。
3. 同时也导致 `debug` 难度陡增，必须在开发时就避免 `bug` 的产生。
4. 编辑器运行中会出现各种边界情况，需要对所有情况进行覆盖。

5.5.3. 应用知识点

1. 模块化程序编写
2. `LibGraphics` 的使用、`Win32 API` 的使用
3. 指针的使用
4. 链表的使用
5. 文件的读取与保存

6. 迪科斯彻算法的灵活运用

5.5.4. 合作记录

本项目没有合作记录，因为整个团队只有一个人。

5.6. 收获感言

这是一个非常非常大的项目，开发过程中有诸多挑战。首先是整个项目只能使用 GNU C99 标准下的 C 语言，在刚开始编写项目的时候给习惯了（相比之下）较为现代的编程语言的我造成了很多麻烦；并且要求使用的 `LibGraphics` 能力非常有限，有很多地方都需要人工调整作图。

虽然是个比较困难的项目，但我还是成功完成了它。有一些编程习惯明显地帮助了这个项目的开发：

1. 在写代码之前，先想好自己要写一个什么东西出来。或者说先确定这个项目的框架。这个有助于评估各个模块所需要的功能，以及为了实现这些功能需要的前置功能；并且也可以帮助我厘清各个模块之间的信息传递，避免不应该的数据共享造成不必要的麻烦。

2. 模块化设计。把可以单独编写的模块提出来（比如被 `EditPage` 和 `SimPage` 共同使用的求解模块），减少重复代码编写，减少整个程序的结构复杂度。

3. 降低耦合度。在 C 语言里面完成这个是比较困难的，但是它值得。尽量降低模块之间的耦合度，互相通过接口而不是特化的方法传递数据；隐藏模块内部的实现细节，因为这些细节不需要也不应该被其他模块访问到。

4. 增量式开发。一次肯定不可能把所有东西都写出来。可以只写完最基础的功能，调试完成后再往上面增加新的功能，以免 bug 堆积而导致调试困难。

5. 规范化的代码。千万不能“今天看不懂昨天写的代码”。

这个项目中使用的数据结构和算法都十分简单，不过这个项目还有一层意义：证明这些数据结构和算法是“实用的”，是可以运用到具体的问题上的。

另外这个项目使用了 `git` 进行代码版本管理，虽然对于一个人的团队它的意义比不上有几个人的团队，但这也是为未来更多更复杂的项目进行准备。

总而言之，这次大作业是对我的磨砺和挑战。而我也很高兴能顺利完成这次挑战，给玛纳霏设计了一个可供使用的不可思议迷宫编辑器。

6. 参考文献资料

1. 人物设定和能力来源于《宝可梦不可思议迷宫 空之探险队》。
2. 宝可梦相关数据来源于神奇宝贝百科：wiki.52poke.com。
3. 图像资源来源于 sprites-resource.com 与 sprites.pmdcollab.org。