基于《猫武士 · 新预言》小说原著的 MUD 游戏设计

面向对象程序设计 M2 报告

nullptr 组

2022 年 5 月 7 日

1 人员信息

组名：nullptr
组员：**略**

2 新的设计构想

在本月撰写代码的过程中，我们对于游戏架构产生了一些新的想法，现将这些想法罗列如下：

1. 在原游戏基础上，我们拟设计两种游戏模式，一种是以表格画出简陋的地图，一种是所有系统信息通过命令行文本进行输出。玩家可以根据自己的偏好自由选择不同的游戏模式，这样便可以使得我们的游戏可以更好迎合玩家的不同喜好。

2. 在输出形式上，我们拟采用 `ncurses` 库进行控制台文字输出。相比于一般的字符串输出，`ncurses` 库支持对文字颜色等属性进行修改，创建菜单、接受方向键输入、可以关闭回显和缓冲等，这便可以使得我们的游戏界面更为美观、具有吸引力。

3. 为了使我们的游戏可以在不同 PC 端上均正常运行，我们拟将显示的字符限制在 80*25（CGA）的尺寸之内。这样，当我们启动游戏调用控制台时，我们的游戏一定可以正确的显示，从而避免因控制台界面不够大从而导致的地图错乱等显示问题。~~理论上甚至可以移植到 DOS 下运行。~~

4. 在原游戏基础上，我们的游戏不但有英文版本，还有英译中版本，玩家可以根据自己的语言习惯在进入游戏时切换语言。同时，我们所有的名词翻译都严格按照猫武官方翻译（参考猫武士灰机），使得我们的游戏内容更贴合原 IP 设定。包括但不限于族群名称、名字前后缀、专有名词（如两脚兽、雷鬼路等）。采用 `MultiString` 类实现，具体翻译可以直接见代码。

5. 采用 CMake 生成 Makefile，同时对 Windows 也做了一些适配，在 Windows 下采用 pdcurses 代替 ncurses。同时解决了两个平台下 curses 中文乱码的问题。

6. 代码规范：类名首字母大写，成员变量和函数首字母小写，统一采用驼峰命名。

3 本月具体进度

3.1 反编译

首先我们需要通过反编译原 Flash 游戏来获得游戏数据和地图生成的逻辑。使用 JPEXS Flash Decompiler 来完成，会自动根据 ActionScript 类似字节码的二进制反编译生成伪代码，方便观察。

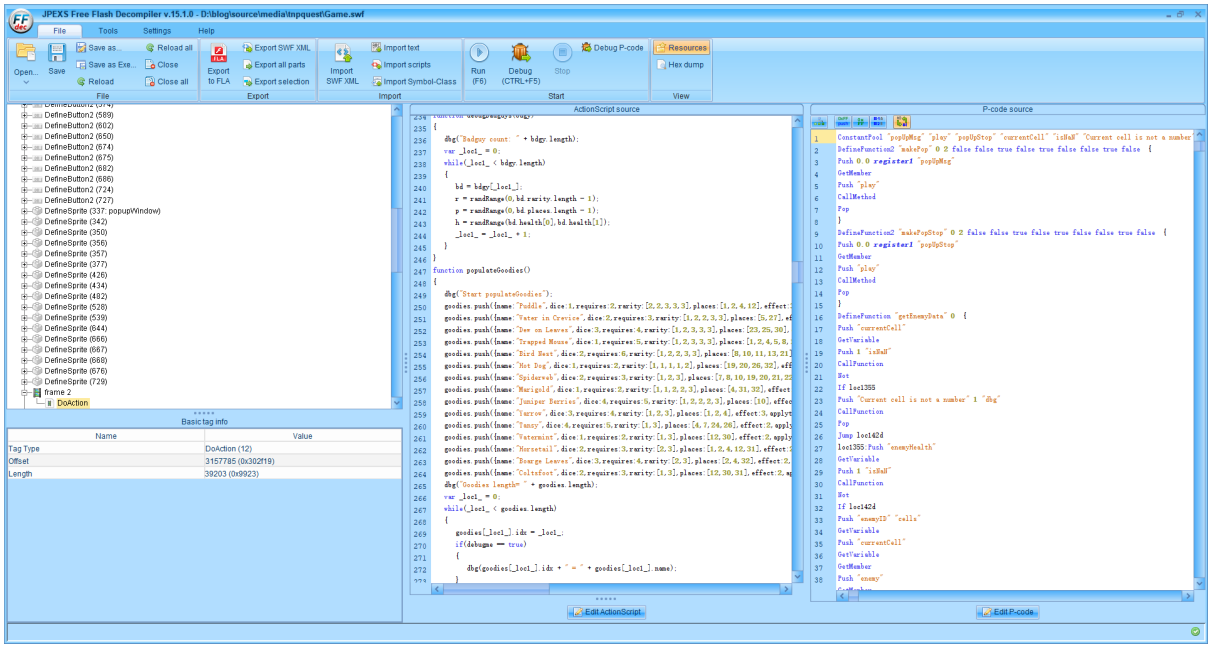

图 1: 反编译截图

3.2 类定义

在本月的工作进程中，我们首先进行了确定游戏的文本内容及各项游戏数值的工作。根据原游戏所提供的数值，我们对这些数据进行了一定的继承与处理，使之符合我们现有的游戏要求。

同时，我们对于游戏总体的框架与数据结构进行了设计，在让游戏基本功能可以顺利实现的基础上尽可能使用面向对象的数据结构进行编程，现对我们目前已设计完成的核心数据结构做如下介绍：

```
// cat.h

Class name: Cat

Private member of class:
    name, clan, health, hunger, thirst, ……: 表示猫的名字、族群、健康、饥饿等等基础
↪   信息等私有属性


Public member of class:
    Cat(), ~Cat(): 构造函数与析构函数
```

```
    setClan(char c), setHealth(int h), setHunger(int h), ……: 设置猫的种族、健康等
↪   各种属性的公开方法
    getApprenticeName() const, getWarriorName() const, ……: 获得猫学徒名字、武士名
↪   字等属性的公开方法
```

Cat 类保存所有角色相关的信息，并且完成角色的各种行为，例如战斗、捕猎、喝水等。

```
//board.h

Class name: Board

Private member of class:
    screens, totalScreens, preys, injuries;
    benefits, enemies, stars, ……: 表示地图上敌人、星星等信息的私有属性
    cells[screens][rows][cols], nowScreen, nowRow, nowCol: 表示目前所在位置在地图中
↪   的屏数、行数、列数的私有属性

Public member of class:
    rows, cols, boxWidth, boxHeight: 地图长度、宽度等基本的公开属性
    Board(std::minstd_rand &rng): 构造函数
    getCell(int board, int row, int col): 获取 cell 信息
    placeItems(std::minstd_rand &rng, std::vector<T> items),
↪   placeTerrain(std::minstd_rand &rng), ……: 对地图进行事件放置等操作的公开方法
    clearScreen(), move(int ch): 清屏、移动函数
```

Board 类保存地图相关的信息，地图由 13 屏组成，每屏 5 行 8 列（原游戏如此）。同时提供了一些方法，以便上层调用来生成地图。

```
//cell.h

Class name: Cell

Private member of class:
    visibility: 表示可见性的私有属性
    *item, *terrain, *star: 指向事件、地形、星星的私有指针

Public member of class:
    Cell(): 构造函数
    *getItem(), *getTerrain(), getVisibility(): 获取格子上的物品、地形、可见性等信
↪   息的公开方法
    setItem(const Item *item), setTerrain(const Terrain *terrain), ……: 设置物品、
↪   地形等属性的公开方法
    renderTerrain(), renderItem(): 渲染格子上的地形、物品的公开方法
```

Cell 类表示地图上的一个格子，每个格子都有一个地形，还可能有物品（包括敌人、猎物等）或族群必需品（也就是星星）。

```
//item.h

Class name: Item

Private member of class:
    name: 表示物品名称的私有属性（双语）
    counts, places: 表示物品数量、位置等的私有属性
    explore, scent: 表示该物品是否可探索，可闻到的私有属性

Public member of class:
    Item(MultiString name, std::vector<int> counts, std::vector<int> places, bool
↪   explore, bool scent): 构造函数
    getName(), &getCounts(), ……: 获取物品名称、数量等信息的公开方法
    trigger(): 触发事件纯虚函数

//defense.h; enemy.h; injury.h; prey.h

Class name: defense, enemy, injury, prey
All public inheritanced from class Item.
```

Item 类是物品的抽象基类，记录了物品生成和使用的基本属性，以及在猫不小心走上去时会发生的事件 trigger()。各种物品实体类直接作为常量硬编码在源代码里，记录初始生成数据。

```
//star.h

Class name: Star

Private member of class:
    name, enemies: 表示星星名字（双语）、敌人信息等私有属性

Public member of class:
    Star(MultiString name, std::vector<int> enemies): 构造函数
    getName(), getEnemies(): 获取星星名字、敌人信息等的公开方法
```

Star 类表示必需品生成相关的属性，其中 enemies 原游戏就没用到，只是为了兼容而保留。

```
//terrain.h

Class name: Terrain

Private member of class:
    name, visible, *cells: 表示地形名字（双语）、可见性、所在单元格等信息的私有属性

public:
    Terrain(MultiString name, bool visible): 构造函数
```

`getName()`, `getVisible()`: 获取地形名称、可见性信息的公开方法
`addCell(Cell *cell)`, `randomCell(std::minstd_rand &rng, Cell *&cell)`: 随机生成并设置单元格信息的公开方法

Terrain 类表示地形，还在生成地图时记录了该地形的所有格子，方便后面放置物品。`visible` 也同理只是为了兼容而保留的，实际都为真。

3.3 游戏初始化

除去上述数据结构的设计之外，在本月的工作中我们还将该 MUD 游戏的初始化部分推进完毕。在 `controller/game.cpp` 文件中，经过代码的撰写，目前我们的程序已可以使用 CMake 进行编译运行，并已经能够进入游戏初始界面，然后进行角色的随机创建。可以说，对于游戏正式开启之前的所有工作，我们都已准备完毕。现对 `game.cpp` 中已实现的功能如下简要展示：

```
//game.cpp

Game::Game(int argc, char *argv[]) : rng(std::time(NULL)), board(rng),
↪  maxStep(5){……}
//构造函数

Game::~Game(){……}
//析构函数

void Game::run(){……}
//启动游戏

void Game::parseArguments(int argc, char *argv[]){……}
//解析参数，向玩家提供展示帮助信息、设置中英文、修改游戏基本设置等功能

void Game::readAttributes(){……}
//读取属性，赋予玩家昵称、族群等基本属性
```

游戏中猫的移动也已经实现，方便调试生成的地图数据。由于影族猫可以斜着走，必须找到一种方法在四个方向键之外增加四种方向。理论上可以采用 Ctrl/Shift/Alt 组合键，但不太自然，最好能识别同时按下两个键，例如同时按下左和上表示左上。目前采用了 ncurses 提供的 `halfdelay` 模式，尝试读取两个按键，如果第二个按键没有超时（阈值为 ncurses 支持最小的 100 ms）就认为两个按键同时按下，走对角线，并且只算走了一步（后面每次走的步数是有上限的）。

3.4 关于可用性的说明

本游戏支持在类 UNIX 环境和 Windows 环境下运行，其中类 UNIX 环境需要自行安装 `libncurses6`，Windows 环境需要自行编译安装 `pdcurses`。在 Windows 下 UTF-8 边框字符和中文字符可能显示有问题，可根据需要加上 `--fallback` 选项。UNIX 环境中文请不要轻易使用。

Windows 控制台有旧版控制台、新版控制台（`conhost.exe`）和 Windows Terminal 三种，后两种应该在 Windows 10 及以上存在，具体可用性列表如下：

环境	语言	可用性
UNIX	英文	原生支持，也可以使用 fallback
UNIX	中文	原生支持，请勿使用 fallback
Windows 旧版控制台	英文	请使用 fallback
Windows 旧版控制台	中文	请使用 fallback
Windows 新版控制台	英文	原生支持有小 bug，可以使用 fallback
Windows 新版控制台	中文	不支持！请使用旧版控制台
Windows Terminal	英文	原生支持，也可使用 fallback
Windows Terminal	中文	不支持！请使用旧版控制台

图 2: 调整 Windows 新旧版控制台

3.5 运行截图

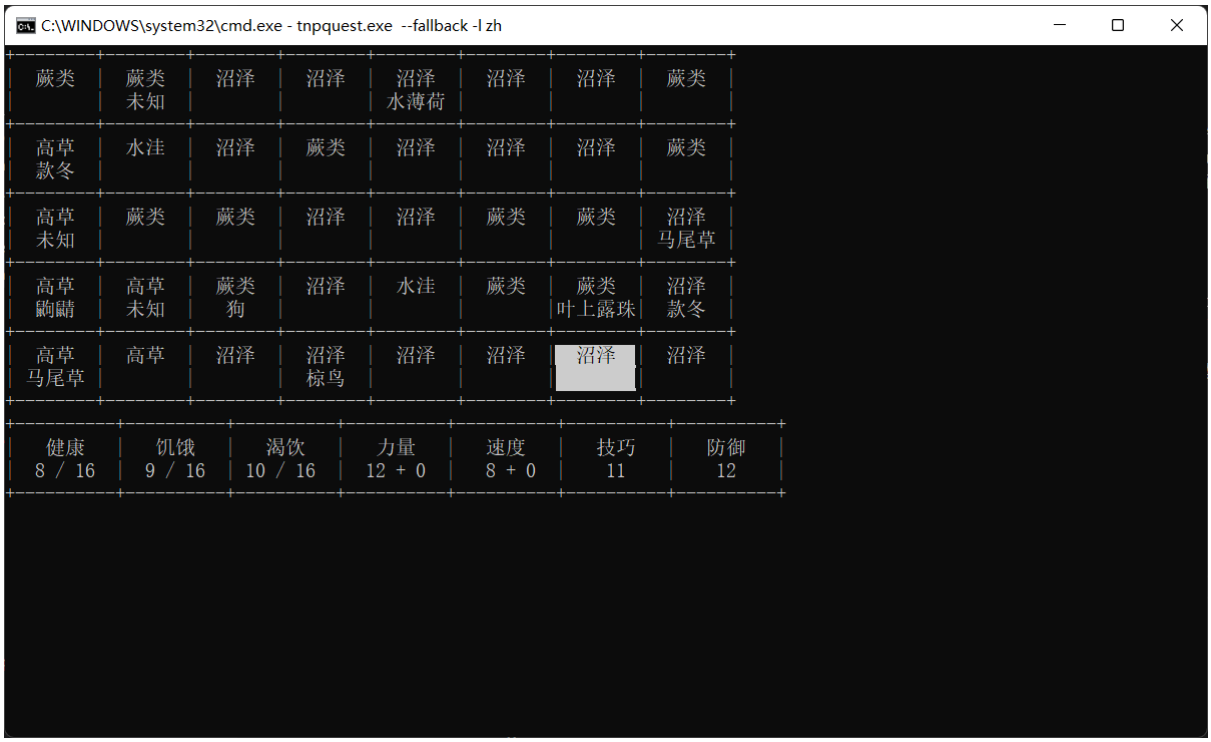

图 3: Windows 旧版控制台

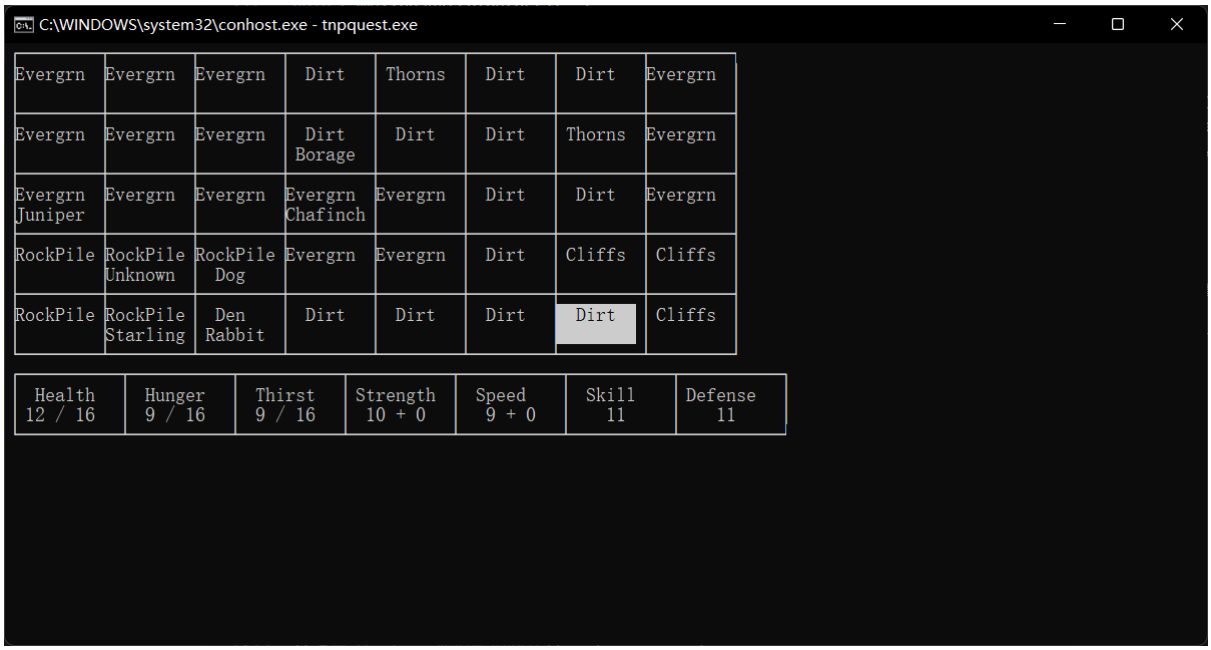

图 4: Windows 新版控制台

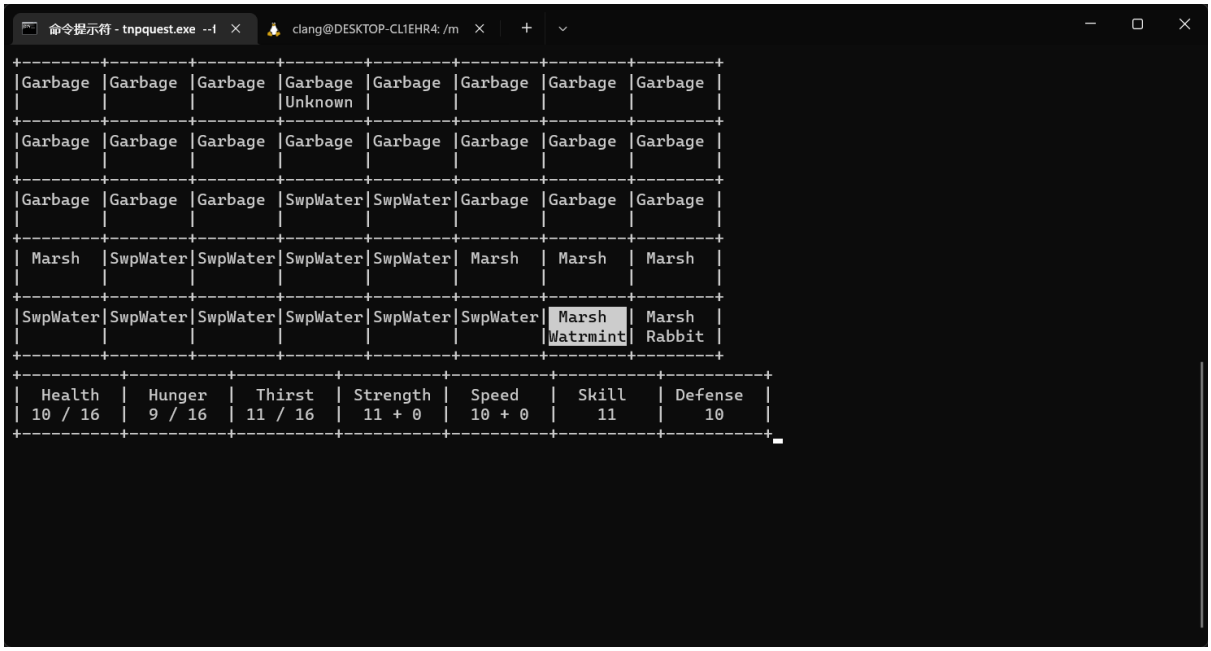

图 5: Windows Terminal

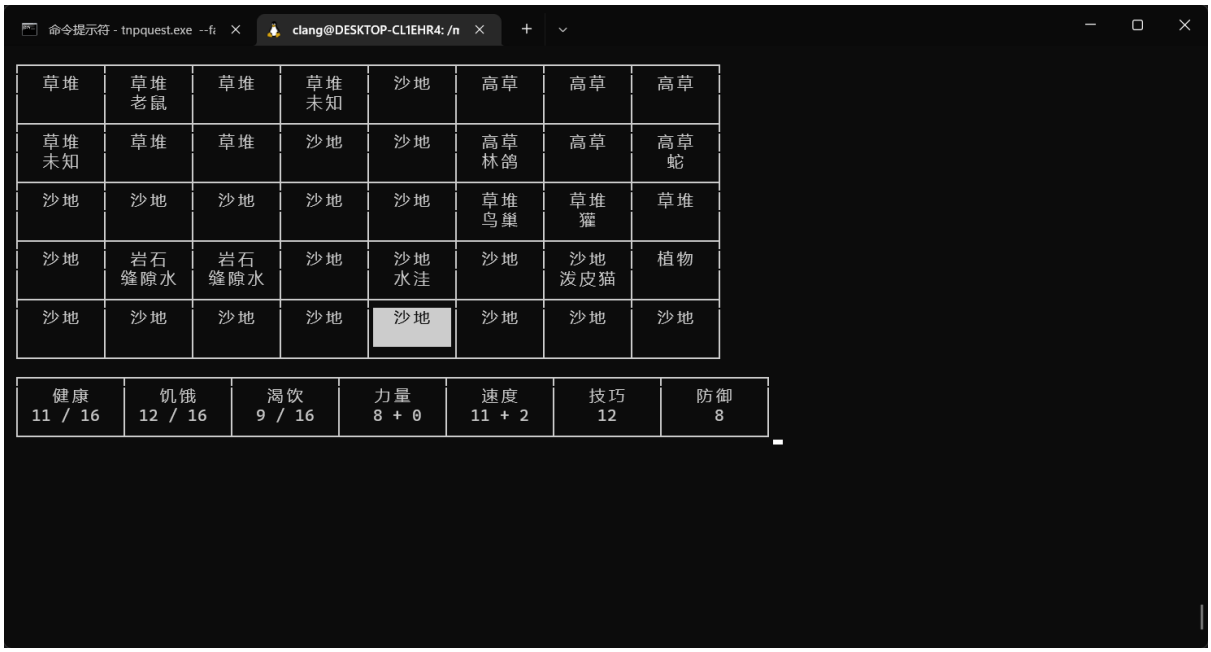

图 6: 类 UNIX (WSL)

4 下周期工作计划

- 撰写游戏开始后主体部分的代码，实现战斗、饮水、触发随机事件等游戏功能
- 完善现有数据结构并创建新的数据结构，或对现有数据结构上使用派生、友元等方式，使得其支持我们新添加的功能
- 对整体代码进行 debug，使之可以正常运行