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 M3 报告

nullptr 组

2022 年 5 月 20 日

1 人员信息

组名：nullptr
组员：**略**

2 本月具体进度

2.1 完成游戏主体代码的撰写

在本周期的工作中，我们基本完成了游戏代码的撰写，使得游戏的主体部分得以正常运行，游戏基本的移动、战斗、饮水、触发随机事件等功能均已完成。现将游戏这些部分的代码做如下介绍：

```
//game.cpp
//pseudo code

bool Game::process()
{
    if (can continue to move) then
        if (input == 'q') then
            quit;
        if (input == '\n' || input == '\r') then
            stop, lose val and go next;
        backup current coordinates;
        if (illegal move) then
            go next;
        if (can move diagonally) then
            put terminal into halfdelay mode;
            if (input == 'q') then
                quit;
            switch terminal into normal mode;
            if (illegal move) then
                rollback and go next;
```

```
        if (trigger event) then
            generate step, lose val and go next;
        if (step <- step - 1 and step == 0) then
            lose val;
    else then
        do action
        if (quit action) then
            quit;
        generate step;
    go next;
}
```

process() 函数主要负责处理猫的移动。判断当前状态是否可以继续移动，如果可以则从键盘读入输入，然后执行移动命令。其中影族可以斜着走是通过短时间内同时按下两个方向键完成的，在上次报告中已经详细介绍，这里不再赘述。若没有剩余步数，则扣除一定的属性值，玩家可以选择六种行动中的一种（见下文），并生成新一轮的移动步数。

```
//game.cpp
//pseudo code

void Game::renderStatus()
{
    if (fallback) then
        print the title of all status;
        print the frame of all status;
    switch (clan of cat)
        case thunder: print "ThundrClan";
        case wind: print "WindClan";
        case river: print "RiverClan";
        case shadow: print "ShadowClan";
        default: print "Unknown";
    print the contents of other status;
    print the frame of all status;
}
```

renderStatus() 函数主要负责渲染属性栏底下的信息框，即将猫的族群、学徒名、地图编号（原游戏没有）、剩余步数、已收集星数等信息展示出来。

```
//game.cpp
//pseudo code

bool Game::doAction()
{
    print the buttons of action;
    choice <- 0;
```

```
    for (;;) do
        for (i <- 0 to 2) do
            for (j <- 0 to 3) do
                if (i * 3 + j == choice) then
                    turn on the A_REVERSE attributes;
                print the selected button in reverse color;
                turn off the A_REVERSE attributes;
        if (input == 'q') then
            quit;
        if (input == '\n' || input == '\r') then
            break from the loop;
        if (input == KEY_UP) then
            move up the choice;
        else if (ch == KEY_DOWN) then
            move down the choice;
        else if (ch == KEY_LEFT) then
            move left the choice;
        else if (ch == KEY_RIGHT) then
            move right the choice;
        print the blank;
        refresh the screen;
    switch (choice)
        case 0: go battle;
        case 1: go check scent;
        case 2: go hunt;
        case 3: go explore;
        case 4: go rest;
        case 5:
        if (no water) then
            MsgBox print "No water here";
        else then
            go drink;
    go next;
}
```

doAction() 函数主要负责对猫的行动进行管理。在控制台的底部，我们会打印出战斗、检查气味、捕猎、探索、休息、喝水六个按钮。玩家可以在六个按钮中选择一个进行行动，而我们的 doAction() 函数则会给出响应的控制。同时，和地图移动一样，我们对当前选择的按钮也进行了反色处理，使得玩家可以清楚看出自己目前位于哪个按钮上。

为了有关类调用方便，战斗、检查气味、捕猎和探索由 Board 类完成，休息和喝水由 Cat 类完成，因为前面几个活动和地图数据密切相关。事实上，战斗可以直接调用 Enemy::trigger 来完成，而检查气味和探索完全是地图操作。捕猎比较复杂，如果玩家直接走到猎物的格子上，会触发 Prey::trigger 而吓走猎物，但如果主动捕猎则可能能捕捉到猎物。

2.2 对类功能的实现与完善

在本月的工作进程中，我们对此前已经定义、但还未完成功能的类进行了功能的填充与改善，目前我们游戏正常运行所需的类功能基本已经齐全。现将这些部分的代码做如下介绍：

```
// cat.h

int Cat::generateStep() const;
// generate step from 1 to speed;if clan is wind, add 2 to the step

void Cat::setMaxStep(int m);
// set maxStep = m

void Cat::loseVal();
// if no more step, lose health, hunger and thirst

void Cat::checkDead() const;
// if not invincible and any status < 0, game over

void Cat::loseHealth(int h);
// set health -= h

void Cat::gainHealth(int h);
// set health += h

void Cat::gainHunger(int h);
// set hunger += h

void Cat::gainThirst(int h);
// set thirst += h

int Cat::rollVal(CatVal val) const;
// roll val from 1 to skill/speed

int Cat::rollAttack() const;
// roll attack from 1 to strength;if clan is thunder, add 2 to the attack

int Cat::rollDefense() const;
// roll defense from 1 to defense

void Cat::flee();
// decrease health, hunger, thirst to flee from battle

bool Cat::hasFled();
```

```
// return fled

void Cat::resetFled();
// reset fled to false

void Cat::incStar();
// increase stars

int Cat::getStars() const;
bool Cat::getDiagonal() const;
bool Cat::getWaterproof() const;
// public method to get the value of star, diagonal and waterproof

void Cat::rest();
// roll to gain 1 or 2 health, show the info in msgbox

void Cat::drink();
// roll to gain 1 or 2 health, show the info in msgbox
```

Cat 类新增了有关移动、喝水、休息相关的公有函数成员，方便实现游戏中的相关功能。同时还增加了显示及修改 stars, diagonal, waterproof 等属性的公开方法，方便在类外对这些私有成员进行读取或修改。

```
// board.cpp

void Board::backupCoordinates();
// record current position into backup ones

void Board::restoreCoordinates();
// set current position with backup ones

bool Board::trigger(Cat &cat);
// trigger events and check if fled

int Board::getScreen() const;
// public method to get the value of nowscreen

void Board::moveTarget(int ch);
// modify the target row and col

void Board::chooseTarget(bool diagonal, bool fallback);
// check keyboard input to choose target, print the target cell in reverse color

void Board::battle(Cat &cat, bool fallback);
```

```
// check if there is an enermy; then check if fled and print the info in msgbox

void Board::hunt(Cat &cat, bool fallback);
// check if can hunt and record current position

void Board::explore(Cat &cat, bool fallback);
// explore and set the visibility

bool Board::getWater() const;
// check if there has water
```

Board 类保存地图相关的信息。在本月的进度中，我们设置的方法主要是为了控制当前位置，以及对地图的周围格子进行战斗、捕猎、探索、喝水等操作。

```
// cell.cpp

bool Cell::trigger(Cat &cat);
// check if there is an enermy to battle and deal with fled

bool Cell::isEnemy() const;
// check if item is enermy

bool Cell::hunt(Cat &cat);
// check if item is prey

const Terrain *Cell::getTerrain() const;
// return terrain
```

Cell 类表示地图上的一个格子，每个格子都有一个地形。在上个周期的工作中我们已经实现了这些，而在本周期的工作中，我们主要为该类添加了触发器以触发战斗事件、对当前格子是否有敌人的判断以及是否可以捕猎的判断；此外，还公开了获取地形属性的方法。

而在 item.h 中，我们在 Item 类中定义了一个纯虚函数 trigger 以供子类使用。而在继承自 Item 的 Enemy, DefenseAction, Injury, Prey, Benefit 几个类中，我们均对该纯虚函数进行了重载。接下来对这些 trigger 进行简单介绍：

```
//triggers from enemy.cpp, defense.cpp, injury.cpp, prey.cpp
//pseudo code

void Enemy::trigger(Cat &cat) const
{
    roll attack and health of enemy;
    first <- true;
    do
        print attack info in msg;
        if (first) do
```

```
            create msgbox;
        else do
            int option <- defend or flee;
            if (option == 2) do
                flee and return;
        first <- false;
        roll defense of cat;
        dmg_to_cat <- max{0, attack_of_enemy - defense_of_cat};
        cat lose health;
        roll attack of cat;
        roll defense of enemy from 0 to defense;
        dmg_to_enemy <- max{0, attack_of_cat - defense_of_enemy};
        enemy lose health;
        print lose health info in msgbox;
    while (enemyHealth > 0);
}

void DefenseAction::trigger(Cat &cat) const
{
    roll damage in damagerange;
    roll result;
    print defense info in msg;
    result <- max{damage - result, 0};
    print health losing info in msg;
    create msgbox;
    cat lose health;
}

void Injury::trigger(Cat &cat) const
{
    roll damage from 1 to damage;
    print max_damage and injury info in msgbox;
    cat lose health;
}

void Prey::trigger(Cat &cat) const
{
    print scared away info in msgbox;
}

void Benefit::trigger(Cat &cat) const
{
```

```
    roll dice to see whether to use the benefit
    if (dice > require)
        apply health/hunger/thirst to cat
        show success in msgbox
    else
        show fail in msgbox
}
```

从上数代码中我们可以看出，无论是遇敌战斗的事件、在战斗中防御的事件、遭遇受伤的事件，还是吓走猎物的事件，其触发器都是基于 `Item` 类的纯虚函数 `trigger` 中实现的。这使得我们的程序实现了动态联编，实现了不同触发器之间接口的统一。这样当玩家在地图上移动时，遇到各种物品，只需要调用 `trigger` 即可实现对应的触发事件。

3 运行截图

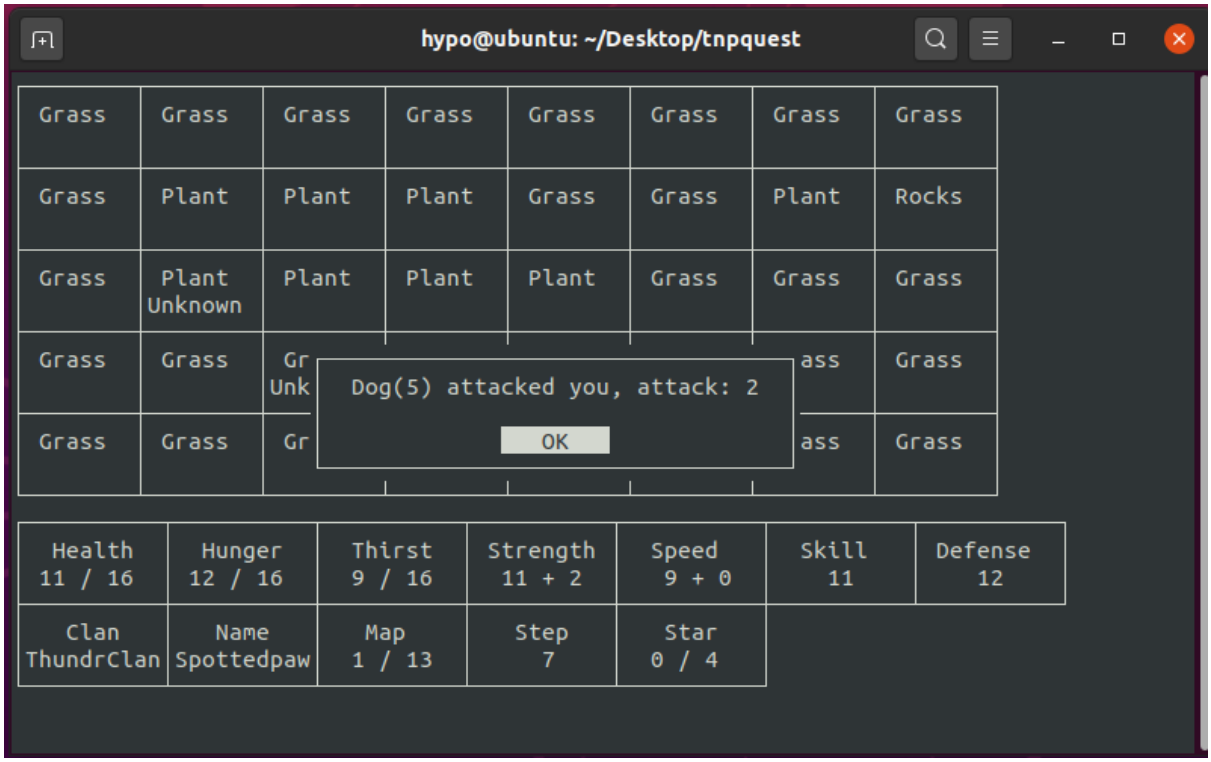

图 1: battle

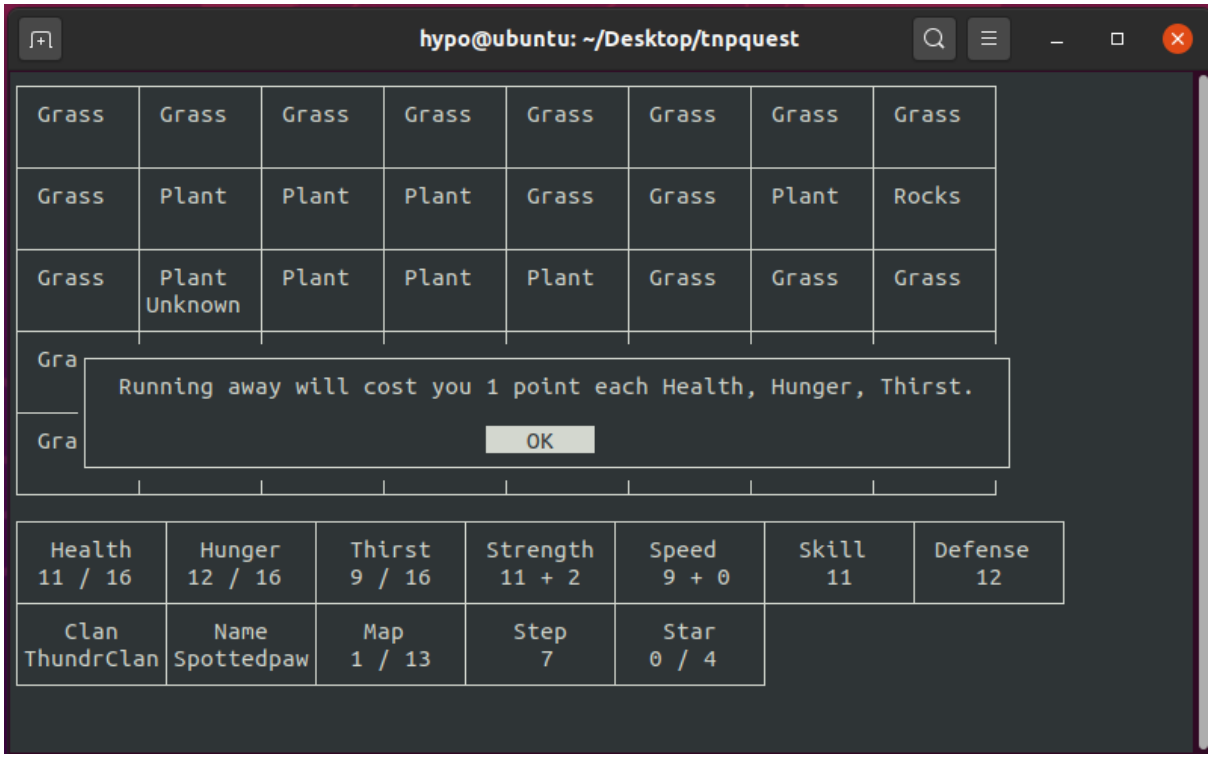

图 2: flee

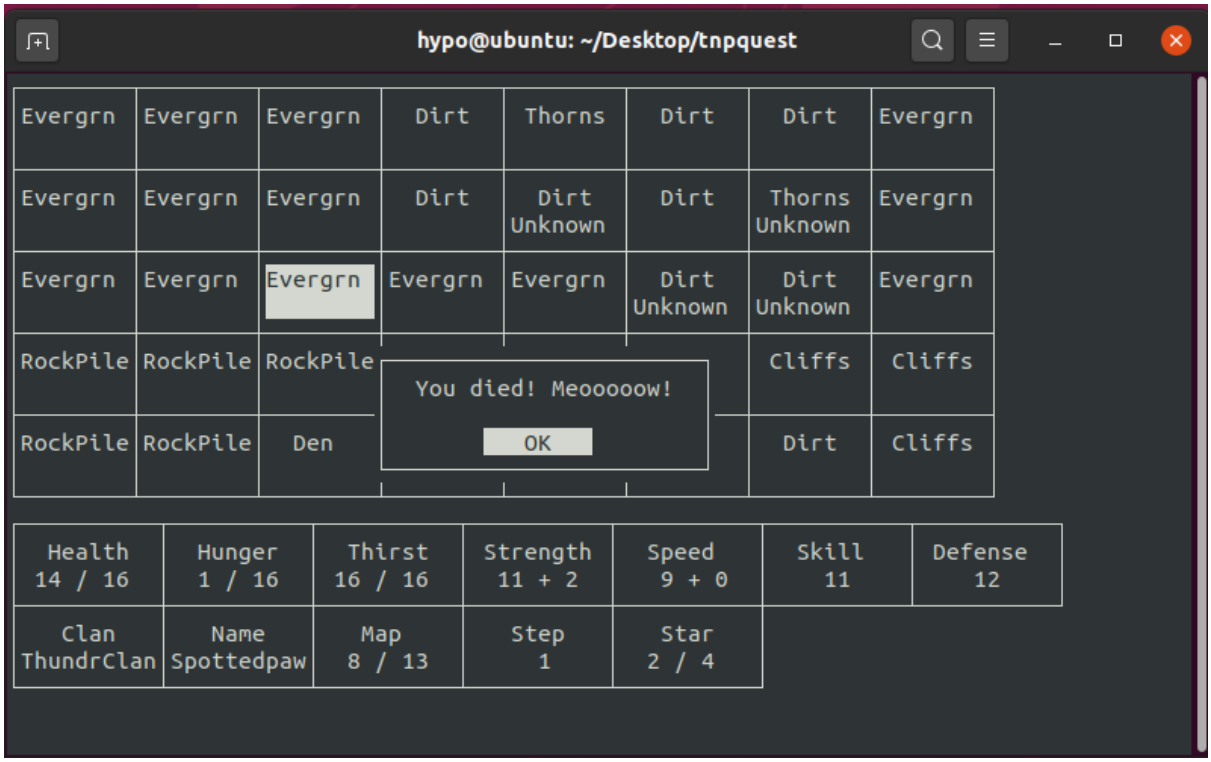

图 3: game over

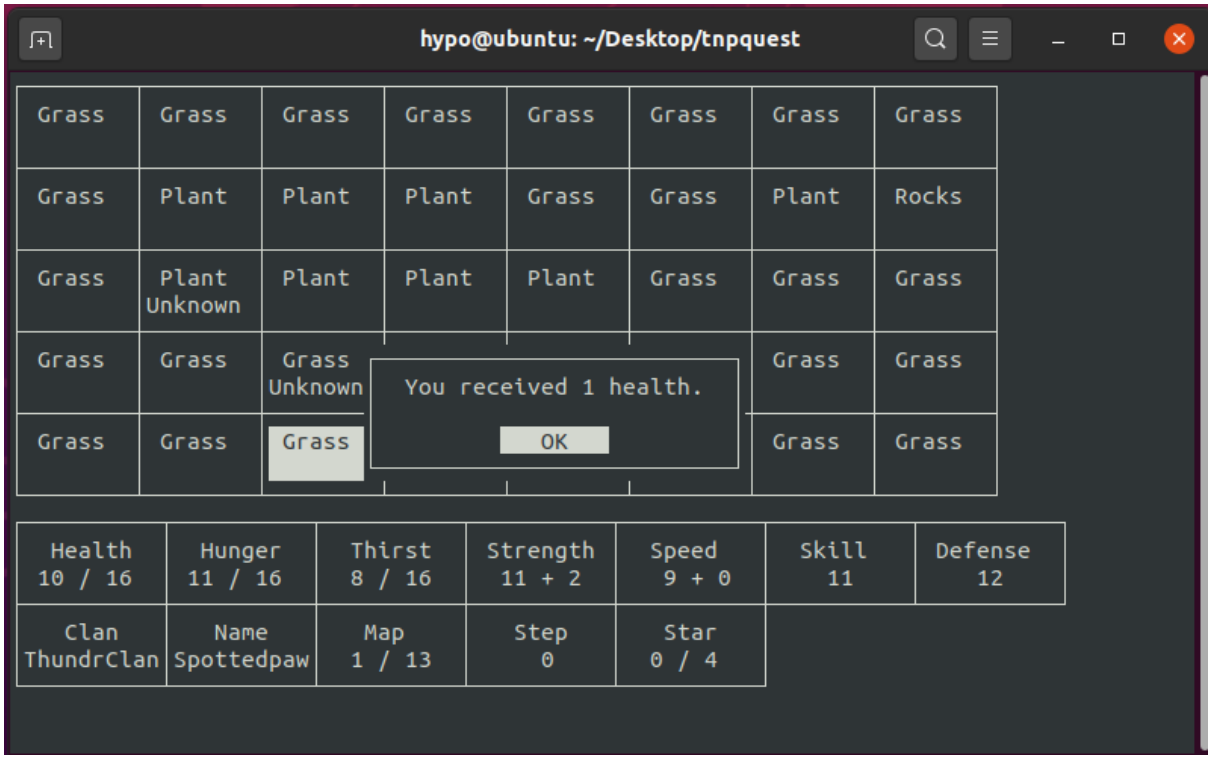

图 4: rest

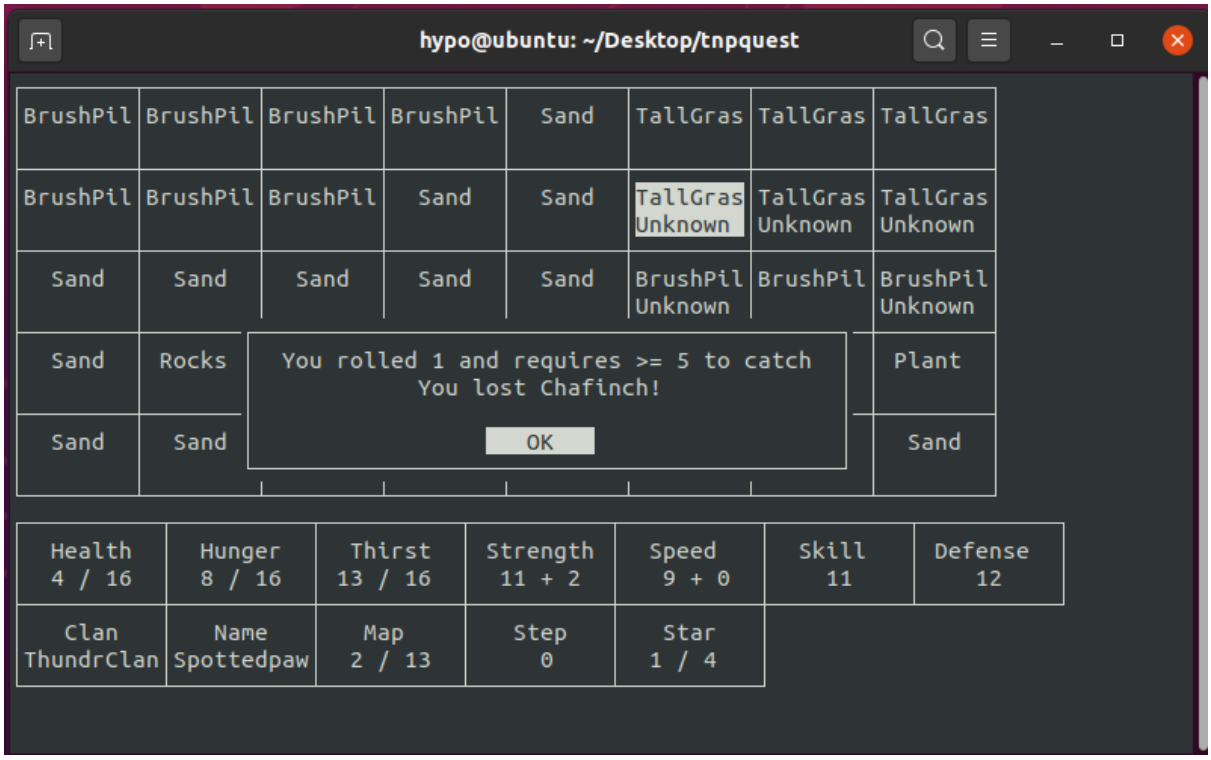

图 5: hunt

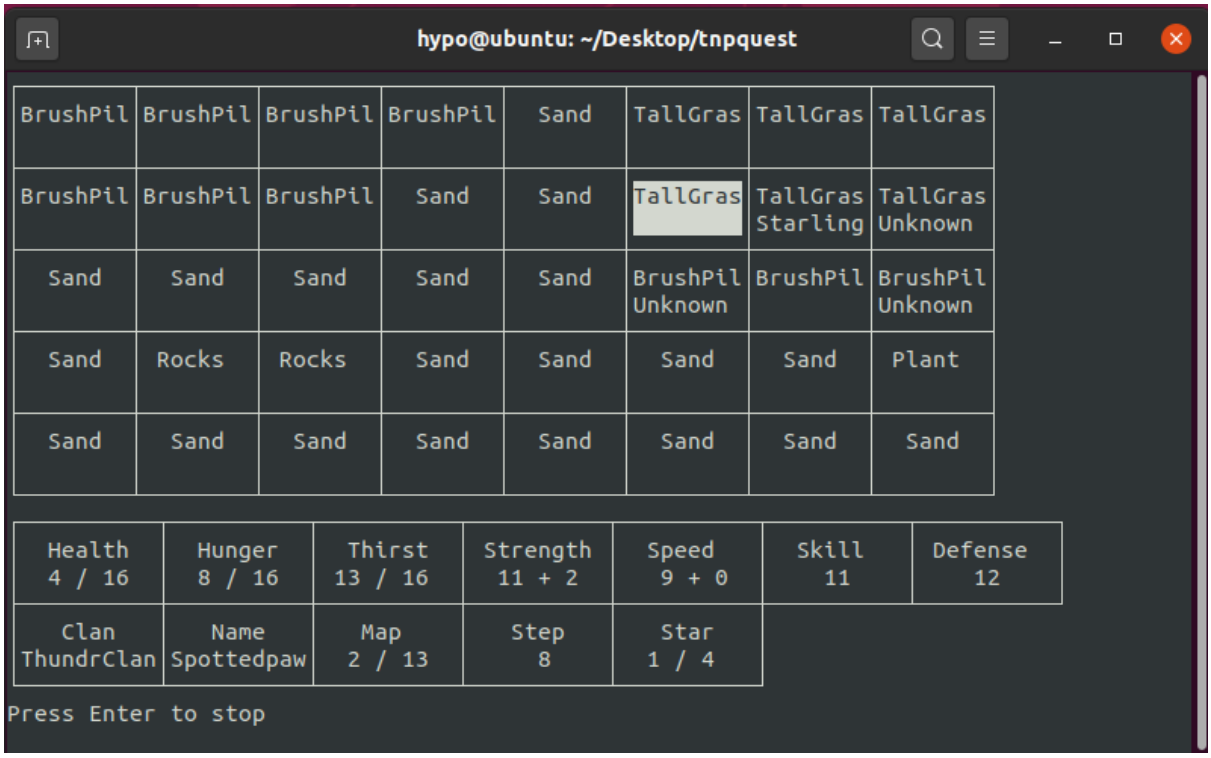

图 6: explore

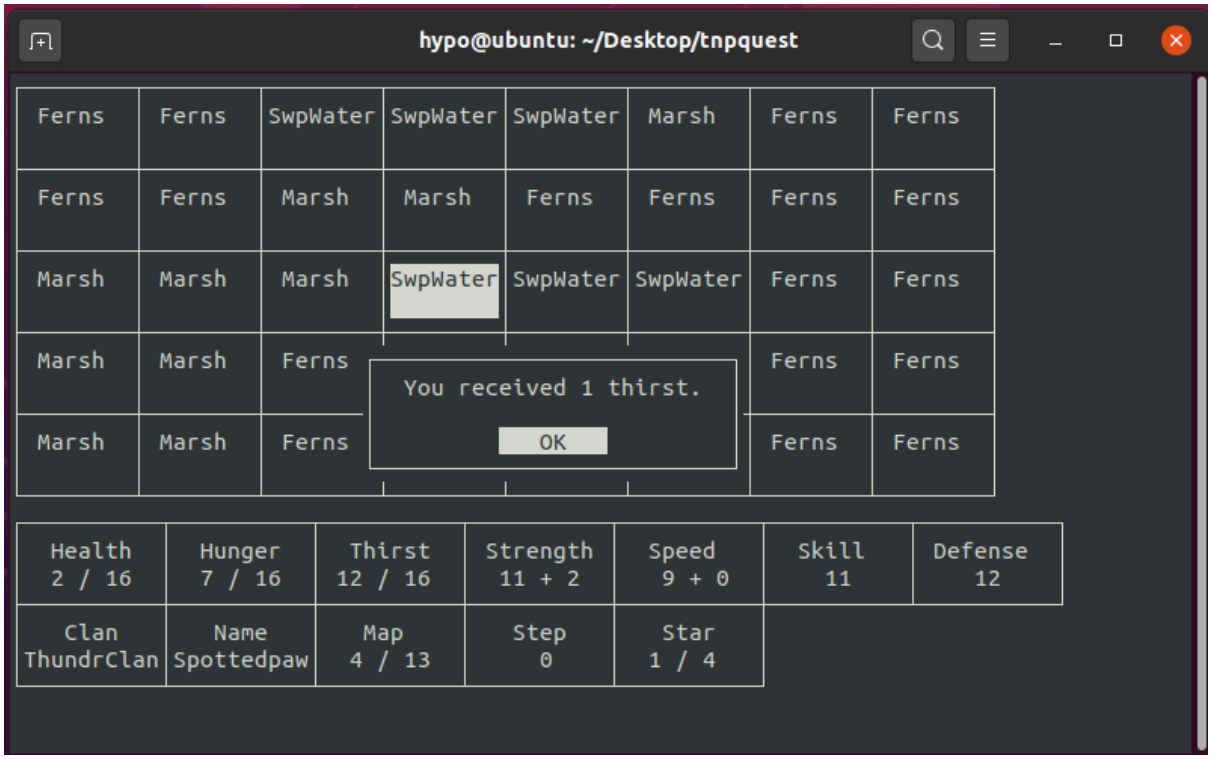

图 7: drink

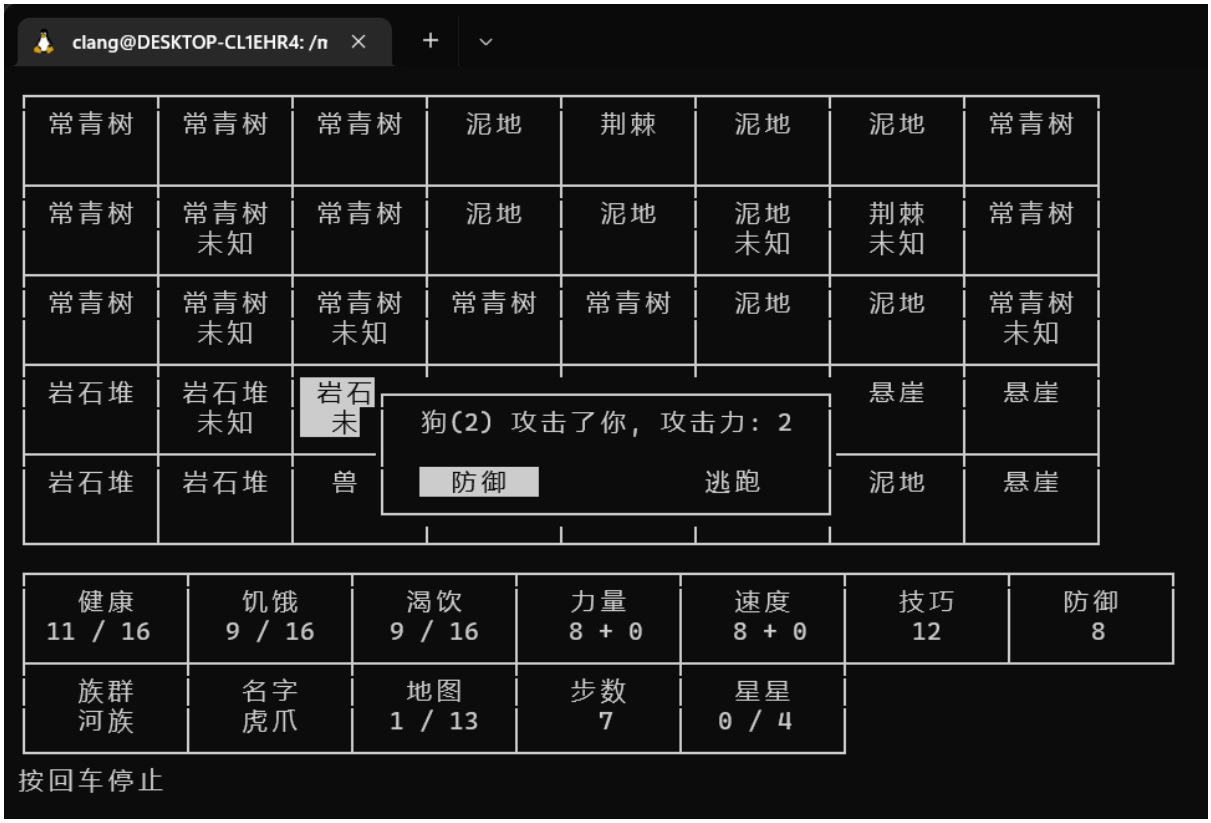

图 8: 中文界面

其中中文界面在 Windows 下仍有很多对齐的 bug，主要是 Windows 终端对 Unicode 支持实在是不佳，可能不会修复。英文无影响。

4 下周期工作计划

- 继续针对代码细节进行 debug，完善有关违规操作的处理方式
- 进行工程整体的收尾工作，撰写报告，完善代码注释等