description

Mages & Goblins

基本介绍

`Mages&Goblins`是一个回合制战棋类游戏。在`M x N`的战场(`Field`)上（`M`为战场高度，`N`为战场宽度），玩家控制法师（Mages）通过移动、攻击和释放法术来消灭敌方的哥布林（Goblins）单位。如果玩家成功消灭了所有的敌方哥布林则玩家获胜，否则失败。哥布林的行动将由AI控制。

基本任务（25分）

- 如果因为地图格式错误无法装载，则输出`Failed to load map!`后退出游戏。
- 我们规定地图的大小不超过`20 x 20`
- 我们提供的地图不会出现单位和特殊地形出现在同一个位置的情况。

在读取地图后，你需要修改`field.cpp`中的`ostream& operator<<(ostream& os, const Field& field)`方法，使得能够正确的显示地图：

- 特殊地形必须有如下显示效果：深水显示为`~~`，高山显示为`/\`。
- 法师单位显示为`@@`，哥布林显示为`PG`。

本任务的输入格式即为地图输入，输出完整打印的地图（或者`Failed to load map!`）即可。

2. 实现我方单位的移动与施法（7分）

2.1 实现我方单位移动

每回合玩家可以操纵所有单位进行移动，

- 每个单位拥有3点行动力，也就是最多可以走3步。
- 每个单位每一步可以到达上下左右4个临近位置。分别用`W A S D`表示， 如下图所示：

```
    W
A  @@  D
    S
```

- 一个格子如果已经有单位存在，那么无法移动到这个格子上。
- 一个格子上如果是特殊地形（高山或深水），那么无法移动到这个格子上。
- 在实现移动命令时，你需要考虑目的地坐标是否超出地图边界。

在每个回合开始时，游戏首先会**打印出当前的地图状态**，之后要求玩家**根据我方单位在加载时的顺序**针对每一个单位输入至多3个方向（如果有输入不是`W A S D`，则忽略该输入）并按下回车进行移动，接着是**敌方单位按照加载时的顺序**移动。在每一个友方单位移动之后，游戏会打印出当前的战场状态，敌方单位的移动**不需要**打印地图状态，每回合开始时打印的地图状态就可以反映上一回合中敌方单位的移动。**在任务2中所有的敌方哥布林不会移动。**

2.2 实现施法

除了移动之外，我方单位可以施放两种法术来攻击哥布林，分别是火球术（fireball）和龙卷风（tornado）。在控制单位行动时时可以在移动命令之后额外输入命令来施放这些法术并产生效果。

一个法师单位在移动命令后（*移动命令可以为空*）输入`X`和方向（`WASD`其中之一）可以从当前所在位置向目标方向施放火球术，其效果为：

- 当无任何障碍物阻挡时，火球将沿着其发射的方向在地图上移动，直到超出地图边界
- 当击中某个单位时，单位将被消灭（无论敌我），火球消失
- 当击中高山时，火球将被阻挡并消散

> 提示：如果一个友方单位被消灭，那么游戏不会再让玩家输入对它的指令。

一个法师单位在移动命令后输入`Y`可以从当前位置施放龙卷风，其效果为：

- 如果本单位周围的8个位置存在其他单位，那么将它们按照从中心出发的方向击退1格， 如下图所示

```
                                TG    TG    TG
  TG TG TG
  TG @@ TG    ---Tornado--->    TG    @@    TG
  TG TG TG
                                TG    TG    TG
```

- 如果击退单位的目标位置为深水（WATER），那么这个被击退的单位会落入深水而死亡。
- 如果击退单位的目标位置为高山（MOUNTAIN）或超出了地图范围，那么这个单位不会被击退。
- 如果击退单位的目标位置为另一个单位，那么另一个单位也会被击退，以此类推。如果这个方向上连续的最后一个单位被成功击退，那么所有单位都被击退1格。否则所有单位保持不动。

2.3 胜利条件判断

在每一个回合**开始并打印出当前地图状态之后**，当战场上只存在一方单位时游戏将终止，你需要实现游戏终止的检查。如果场上只存在我方单位时输出“You Win!”, 只存在敌方单位时输出“You Lose!”。

3. 实现巡逻型敌人（Patrol Goblin） （7分）

巡逻型哥布林的输入**改为**如下格式：

```
R C PG D L
```

前两个是该哥布林的初始坐标，第三个输入`"PG"`仍然是一个固定的字符串，代表这个单位是一个巡逻型哥布林，最后两个输入`D L`分别是初始时朝向的方向（只可能是`WASD`四种）以及每回合该哥布林的行动力。最简单的情况（没有碰到特殊地形以及玩家单位）时，巡逻哥布林在每回合下会朝着一个方向，消耗`L`点行动力移动`L`步，然后反转方向。

3.1 巡逻和攻击策略

具体来说，巡逻型哥布林每回合下会重复如下步骤直到行动终止：

- 哥布林在行动前会根据下面的条件（按优先级从高到低）来决定是否终止该回合的行动：
  - 如果该哥布林上下左右四个方向存在玩家单位，则哥布林会对**上下左右**四个方向进行攻击并**反转**方向，然后终止该回合行动。**注意**哥布林的攻击不会误伤己方单位。
  - 如果当前回合的行动力已耗尽，则**反转**方向，并终止该回合行动
  - 如果前进方向的下一个坐标是这些情况的一种，则**反转**方向，并终止该回合行动：到达地图外，不是平原地形（不能走到水和高山地型上），该坐标已经有了其他单位
- 否则，消耗一点行动力并移动到当前朝向的方向的下一个坐标

当有多个哥布林时，它们的行动顺序按照加载时的顺序，所以有可能出现前一个哥布林移动到一个哥布林的前面，导致这个哥布林反转方向（下面的例子会进行演示）

4. 实现追踪型敌人（Tracking Goblin）（5分）

我们将实现追踪型哥布林，它能够在每回合下移动到**视线内**最近的一个玩家的相邻单位（不能移动到玩家上），并对之进行攻击。

4.1 追踪和攻击策略

哥布林的追踪策略是寻找**视野内**距离最近的一位玩家单位，并计算到达这个玩家单位的**最短路径**，然后在这个路径上移动一定距离（该距离为`min(最短路径长度 - 1)`，最短路径长度减一是因为不能移动到玩家单位上，只能移动到玩家单位的周围）。哥布林移动后，会对它的**上下左右**四个方向进行一次攻击（攻击不会误伤己方单位）。如果视野内没有玩家或者视野内的玩家没有任何一位能够到达，则该回合不动。

哥布林的视野覆盖范围直观上可以理解为上图中的菱形区域，严格的定义为：对于哥布林的坐标`(r,c)`，它的视野范围为`s`的区域中的所有点（用`(x,y`表示）需要满足`abs(r-x)+abs(c-y) <= s`，`abs`为取绝对值函数。

对于两个坐标分别为`(row1,col1)`和`(row2,col2)`，`coordOrder(row1,col1,row2,col2)`返回true当且仅当`row1<row2`或者`row1==row2 && col1<col2`，我们称`(row1,col1)`在坐标偏序上比`(row2,col2)`小。当有多个最近玩家时，我们选择坐标偏序最小的玩家单位。

测试游戏

我们使用`judger.py`脚本做最终的测试，为此你需要将主函数改为**从`cin`中读取地图及之后的用户命令**，然后将结果输出到`cout`。每个任务我们准备了对应的测试案例，放在`data`文件夹中。**你的程序必须通过所有测试案例才能拿到对应任务的满分**。每完成一个任务，你需要将`Mages&Goblins`目录下的代码拷贝到`source`目录下对应的任务文件夹中。特别注意本次作业使用`StanfordCppLib`，因此需要将编译StanfordCppLib产生的cs1604文件夹的**绝对路径**复制到`source/cs1604.txt`下，以让`judger`成功编译你的程序。

```
source
|- 1_task1
|   |- main.cpp
|   ...
|
|- 2_task2
|   |- main.cpp
|   ...
|
|- 3_task3
|   |- main.cpp
|   ...
|
|- 4_task4
|   |- main.cpp
|   ...
|
|- cs1604.txt (include the StanfordCppLib)
```

然后在Windows命令行中运行

```
python judger.py -1  // 1 代表第1个任务，同理可测试2、3、4任务
```

如果测试通过，输出结果

```
[T1 c1] Correct
[T1 c2] Correct
...
```

如果测试不通过，则会显示输出不对应的地方。为了测试所有的结果，可以直接调用

```
python judger.py
```

隐藏测试（5分）

本次大作业有一部分隐藏测试用来测试**所有任务都完成的程序**各种可能出现的极端情况，**通过所有隐藏测试才可以得到满分**。该隐藏测试将不会透露给学生，所以请特别注意自行测试各类边缘情况。自行测试的结果可以和参考程序相对比，参考程序在`demo`文件夹下面。有2个版本：

- `demo1.exe`从`map.txt`中读取地图文件，然后和用户通过**标准输入输出**进行交互。
- `demo2.exe`从`1.in`中读取地图文件和所有用户输入，将结果输出到`1.out`。

前者用来做交互测试，后者用来做文件输入输出的对比测试。此外，调用demo时加上参数`-D`，将进入Debug模式，会显示更多信息。具体使用方式为在`demo`文件夹下打开命令行并执行`.exe`文件即可。

测试案例中程序终止的规范

在交互式游玩（通过`cin`和`cout`执行）时程序只会在一方单位被完全消灭时终止。但在使用案例进行测试时可能出现因为输入流结束而终止的情况，为了在这种情况下你的程序行为和参考程序一致，请不要更改我们在`engine.cpp`中给出的`play`函数中的基本结构：

```
void play(Field& field, istream& is, ostream& os, Vector<Unit*>& units)
{

    int numTurns = 1;
    while (is)
    {
        os << "Turn " << numTurns << " begins:" << endl;
        // Print the new map
        os << field << endl;

        // Check winning
        // Fill in your code here

        // unit moves
        // Fill in your code here

        // Preventing loop in empty program
        string line;
        getline(is,line);

        numTurns++;
```

```
    }
}
```

在上述结构中对于输入流`is`的检查只出现在`while`循环刚刚开始的时候，因此当样例输入文件中给出的指令全部执行完毕之后，当前回合内剩余的没有得到操作命令的友方单位视为**不执行任何行动**，程序在当前**回合结束**时结束`while`循环并终止。另外，如果输入文件**恰好**对于每一个友方单位都给出了命令，那么程序可能会由于输入流中还存在文件结束符`EOF`而进入下一个回合，在下个回合中所有友方单位都不执行任何行动，**回合结束**时程序终止，例如`task2`的第一个样例。总之如果测试案例的输入没有使得游戏分出胜负，那么程序会在回合结束时终止。

为了避免考虑上述的边缘情况，请保持上述`play`函数的基本结构，同时对于每个友方单位请使用`getline()`函数读取其输入，不增加任何额外的输入流判断。遵守上述规范后，你的程序在交互式执行和文件流执行时的行为都会和参考程序相同。

提交文件格式

你需要提交的文件结构应该类似如下形式：

```
<your student number>.zip
|- 1_task1
|   |- main.cpp
|   ...
|
|- 2_task2
|   |- main.cpp
|   ...
|
|- 3_task3
|   |- main.cpp
|   ...
|
|- 4_task4
|   |- main.cpp
|   ...
|
|- cs1604.txt (include the StanfordCppLib)
```