推箱子游戏的搜索

2018011444 自81 魏欣然

1 推箱子游戏简介

推箱子，即sokoban，是一款起源于日本的智力游戏。

游戏在一个长方形棋盘上进行。棋盘格的地形分为地面/墙/墙外。地面上会有数个箱子，以及相匹配数量的目标点。玩家的目标是操控一个小人将每个箱子都推到一个目标点上。

推箱子的规则是：箱子只能推不能拉。箱子后面有墙和其他箱子时推不动。玩家只能向前后左右四个方向进行移动，每次移动一格。

2 游戏向搜索问题的转化

2.1 状态

推箱子游戏可以有不同的棋盘设置，从而有丰富多彩的变化。当针对一局游戏进行搜索时，认为棋盘设置，即地面/墙/墙外/目标格点的位置都是不变的。小人和箱子的位置是可变的，因此"状态"就包含小人的位置和各个箱子的位置。

2.2 起始状态

如上所述，起始状态就是小人和箱子的初始位置。

2.3 终止状态

根据游戏目标，终止状态是每个箱子的位置都和一个目标格点的位置相等的状态。

2.4 状态扩展方式

根据游戏规则，小人仅能向前后左右方向一次移动一格。某些情况（旁边是墙/箱子推不动）无法移动。因此一个状态扩展出的新状态个数不大于4.值得注意的是，由于箱子只能推不能拉，因此小人返回当前状态的上一状态的位置时可能是一个新的状态（在推动了箱子的情况下）。

3 搜索算法

我采用了剪枝优化的A*算法，以在较快的情况下得到最优解。

3.1 局面估计函数h*(n)

A*算法的核心是，在任何一个状态下对该状态到目标状态的距离进行估计。保证该估计值恒小于等于真实值的情况下，A*算法必能搜索到最优解。

我采用的估计函数是：将目前状态所有箱子和目标位置进行一一匹配。匹配后每一对箱子-目标的曼哈顿距离相加即为h*(n)。

该估计值一定小于真实值。h*(n)代表小人推着每个箱子沿曼哈顿距离到达目标的步数。但一方面，小人的位置变化是连续的，推完一个箱子至少还要移动到下一个箱子旁边。中间可能因为路径阻塞要进行多次箱子位置移

动。另一方面，箱子到目标的曼哈顿路径往往有墙阻挡而走不通。因此可以证明h*(n)≤h(n)

3.2 匹配算法

上述估值函数中，有一步是将所有箱子和所有目标点进行匹配。由于我们需要估计值尽量小，因此希望求出最优匹配。

对于有n个箱子的状态，所有匹配情况共n!种。采用暴力法遍历所有匹配，对每种匹配计算曼哈顿距离之和，总复杂度为O(n*n!)。这是一个指数复杂度。

为优化这一步计算，我采用了二分图最优匹配的匈牙利算法。该算法可以在O(n^3)时间内得到最优匹配。该算法使用了开源代码。

3.3 死亡状态

推箱子游戏有着其独特的特点，针对问题本身进行剪枝优化，可缩小搜索空间，提高效率。

对于推箱子游戏而言，一个箱子不能走入非目标格点的墙角。墙角的定义是：一个地面格，其前后左右四邻格中，至少有两个相邻邻格为墙壁。相邻邻格指左前，前右，右后，后左四种情况。如果有箱子处于这种状态，它就一定无法到达目标格点，因此可以判断这种状态是“死亡”的，无需进行继续扩展。

3.4 搜索算法

算法即按照课上所讲的A*算法进行。

```
初始：将初始状态放入Open表（通过堆实现）
搜索过程：
    从Open表中取出f*(n)最小的状态S放入close表
    若该状态为终止状态：
        停止搜索，回溯获取路径，退出搜索
    若该状态为死亡状态：
        什么都不做，进行下一轮搜索
    否则：
        对该状态进行扩展，得到不多于4个子状态
        对于每个子状态C：
            若C是S的父状态：
                什么都不做
            若C在close表中：
                尝试更新其f*(n)
                若更新成功：
                    将其从close表取出重新加入open表
                否则：
                    什么都不做
            若C是新搜索到的状态：
                将g(C)更新为g(S)+1
                将C放入open表中
            若C在open表中：
                尝试更新其f*(n)
                若更新成功：
                    将open表重新排布
                否则：
```

```
                    什么都不做
            循环结束
    循环结束
```

4 程序的安装和使用

4.1 安装

本程序仅支持windows 10操作系统。

在powershell中进入AppPackages/MyPushBox_1.0.1.0_Test文件夹

```
cd AppPackages/MyPushBox_1.0.1.0_Test
```

运行Install脚本

```
./Install.ps1
```

即可完成安装。

安装过程中，可能会出现如下报错：

```
无法加载Install.ps1，因为在此系统上禁止运行脚本
```

对此的解决方案是：

1. 在菜单栏搜索框中搜索“powershell”
2. 找到系统powershell程序，右键单击，选择“**以管理员身份运行**”
3. 输入以下命令，选择“Y”

```
Set-ExecutionPolicy RemoteSigned
```

4. 重新运行Install.ps1脚本

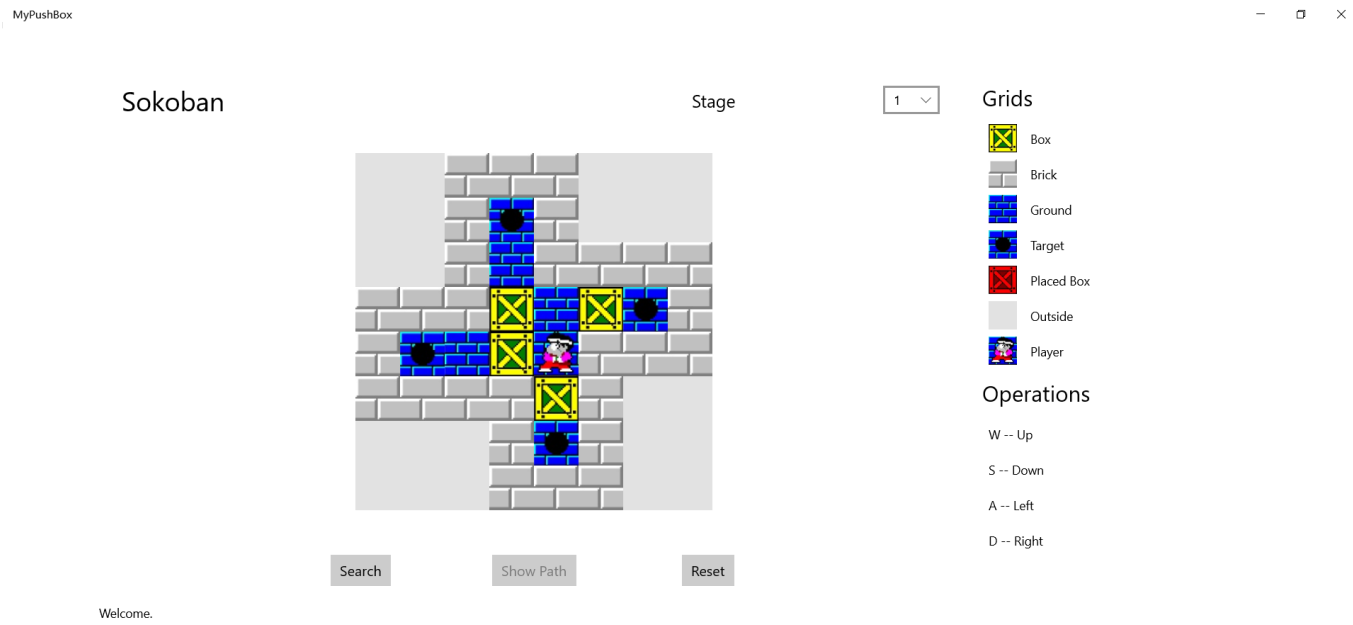
此时会自动弹出系统设置界面。您应选择“**使用开发者模式**”选项。请您放心，该程序是安全且干净的，不会对您的电脑进行任何破坏。在安装完毕后，您可以将此设置项恢复默认状态。

若仍然安装失败，请确保使用开发者模式已打开，并重新运行Install脚本，即可完成安装。

安装后，您可以在**开始菜单**（按windows键）中找到**MyPushBox**应用，单击即可运行。

4.2 使用

应用程序界面如下图所示(需要最大化窗口)



屏幕左侧为游戏区，右侧为图例和按键操作。

手动

当您希望手动探索游戏时，您可以使用键盘上的W S D A四个键进行方向操控，图中的小人会根据您的指示进行移动。

复位

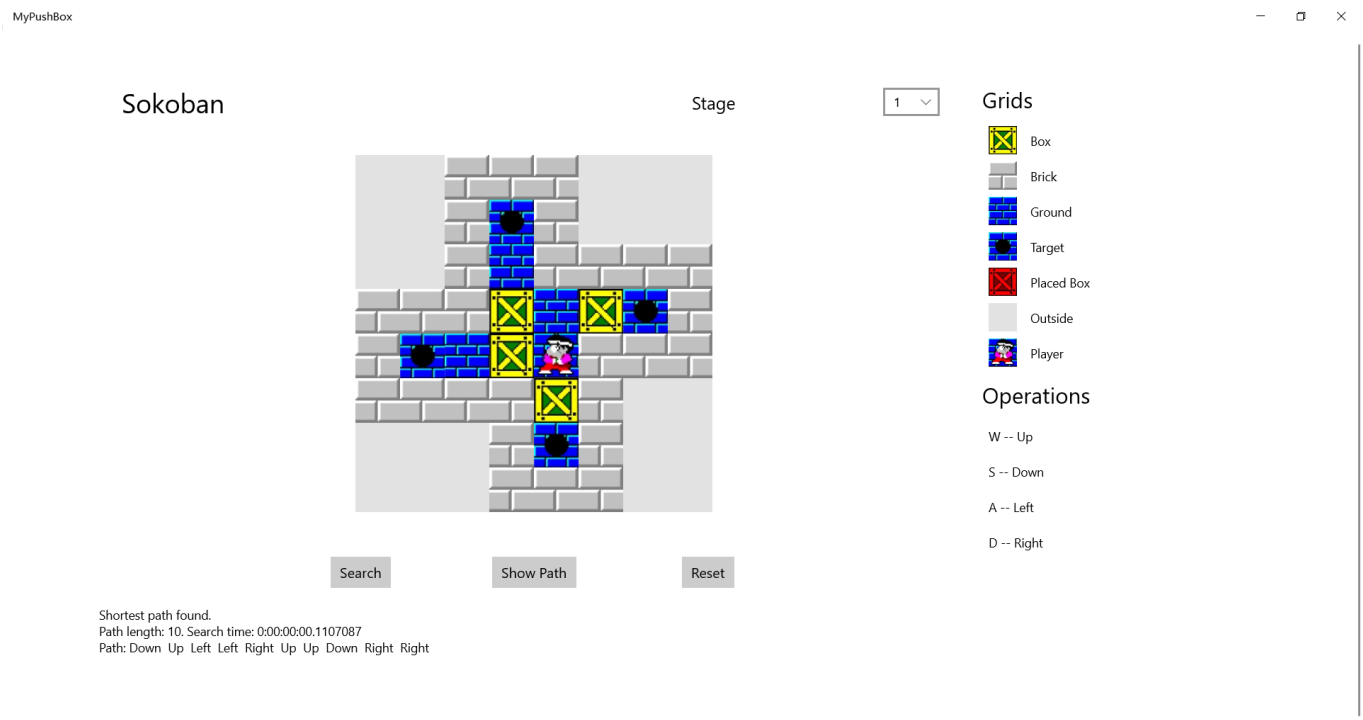
点击Reset按钮即可使游戏恢复初始状态。

搜索

点击Search按钮，算法开始自动搜索最优路径。

根据游戏难度不同，搜索时长不同，最长可能达到30-40秒。过程中请您耐心等待。

搜索完成后，底部显示区域会显示搜索耗时，最优路径长度和最优路径信息。如下图。



自动展示

完成一次搜索后，Show Path按钮将变为可用。此时点击此按钮，将会展示搜索到的路径。

展示完成后，您可以利用Reset恢复初始位置。

关卡

游戏设计了8个不同难度的关卡。难度不是按照关卡号递增。您可以在右上角下拉框中选择不同关卡进行体验。

注意：**更换关卡后，此前的搜索结果不会保存。**

4.3 开发相关信息

本程序采用C#语言，利用UWP(Universial Windows Program)平台进行开发。平台使用windows 10 原生UI组件，交互方式友好，开发过程简单便利。

作业截止时间后，本项目将在github上开源. 地址：https://github.com/weixr18/MyPushBox