slide 滑块拼图游戏说明

滕飞 (Colin Teng)

最后更新日期: 2023 年 9 月 28 日

1 游戏介绍

slide 是一款 linux 平台下的滑块拼图游戏, 在中国习惯被称为华容道, 使用 c 语言开发. 将一副图像按笛卡尔网格拆分为 $m \times n$ 个方格. 并摘除其中一个方格, 记为空格. 游戏者每次可将空格四周的其中一个方格的内容与空格的内容进行交换, 即完成一次滑动操作. 初始时所有方格的内容被随机打乱为任意某个方格的内容. 游戏者的目标是通过逐步的上述滑动操作让每个方格的内容恢复到原始的状态.

需要注意的时, 如果想保持空格的位置不变, 上述操作所造成的方格内容的交换必然为偶置换, 经过上述变换操作的所有可能的状态为全体偶置换群. 因此初始随机打乱内容时, 必须保证打乱的排列与原始排列之间构成偶置换. 如果构成奇置换, 则游戏者永远无法经上述操作恢复到原始状态.

游戏按前后台模式设计. 后台负责游戏规则实现和流程控制, 前台负责与用户交互. 前端开发者可以基于软件包内的游戏引擎部分开发不同的用户界面. 本软件包共包含 3 个部分. 分别为

1. 游戏引擎: 源代码为 sp.h 和 sp.c
2. ncurses 的前端: 源代码为 sp_cs.c
3. gtk3 的前端: 源代码为 sp_gtk.c

其中第 1 部分为游戏的后台控制引擎, 整体游戏流程和规则均由该部分所控制. 第 2 部分为基于 ncurses 库实现的一个终端上的游戏前端用户界面, 适宜在终端环境下运行游戏. 第 3 部分为基于 gtk3 库实现的一个图形用户界面, 适宜在安装有 gtk3 图形界面库的环境下运行游戏.

2 编译运行说明

首先, 编译本软件需要依赖 ncurses, gtk3 和 pkgconfig. 第 1 节所述的 3 个部分的编译运行方法分别为

1. 游戏引擎 (无法直接运行)

```
make
```

2. ncurses 的前端

```
make sp_cs
sp_cs
```

3. gtk3 的前端

```
make sp_gtk
sp_gtk
```

图 1 为 gtk3 界面示例. 用户需要先点击下方图片处的按钮选择一张 jpg 图片作为拼图的原始图像, 并在格数中输入每个维度被分割的网格数, 然后点击开始按钮即可开始游戏. 游戏开始时, 原始图像中位于右下角格子内的图块会被删除, 空格会被置于此处. 游戏中, 点击空格周围的方块可实现一次滑动操作. 点击和空格位于同一行或同一列的方块可一次实现多次滑动操作 (即滑动一行或一列). 完成拼图后, 界面会提示用户的滑动次数.

在软件包的 picture 目录下包含几张用于 gtk3 界面的测试图片. 所有测试图片均来源于互联网, 本软件作者对任何图片都不享有版权. 图片仅用于测试本软件, 不属于本软件的一部分.

图 2 为 ncurses 界面示例. 用户可通过类似 vim 中的 h (左), j (下), k (上), l (右) 键或方向键来移动坐标到某个方格, 按空格键可完成一次滑动, 按 q 键退出. 窗格内显示的数字为该位置所包含图块的原始坐标. 初始时, 位于右下角的图块会被删除, 以便使空位被置于此处.

3 引擎数据结构

游戏的后台运行状态全部保存在结构 `SPDriver` 中.

```
typedef struct strSPDriver SPDriver;
struct strSPDriver
{
      int nx;
      int ny;

      int eloc;
      int celoc;
```

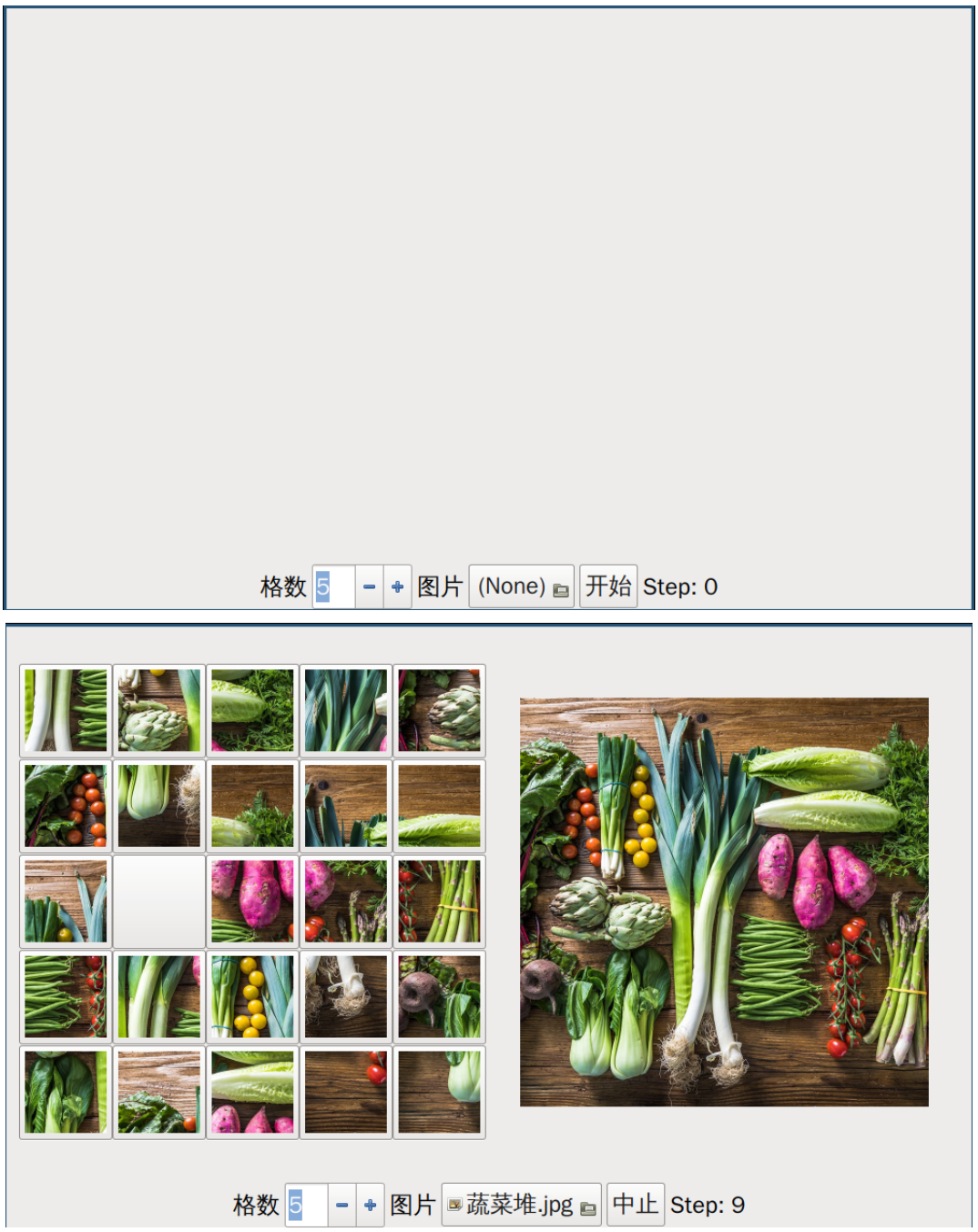

图 1: gtk3 界面示例: 上: 游戏开始前. 下: 游戏进行中

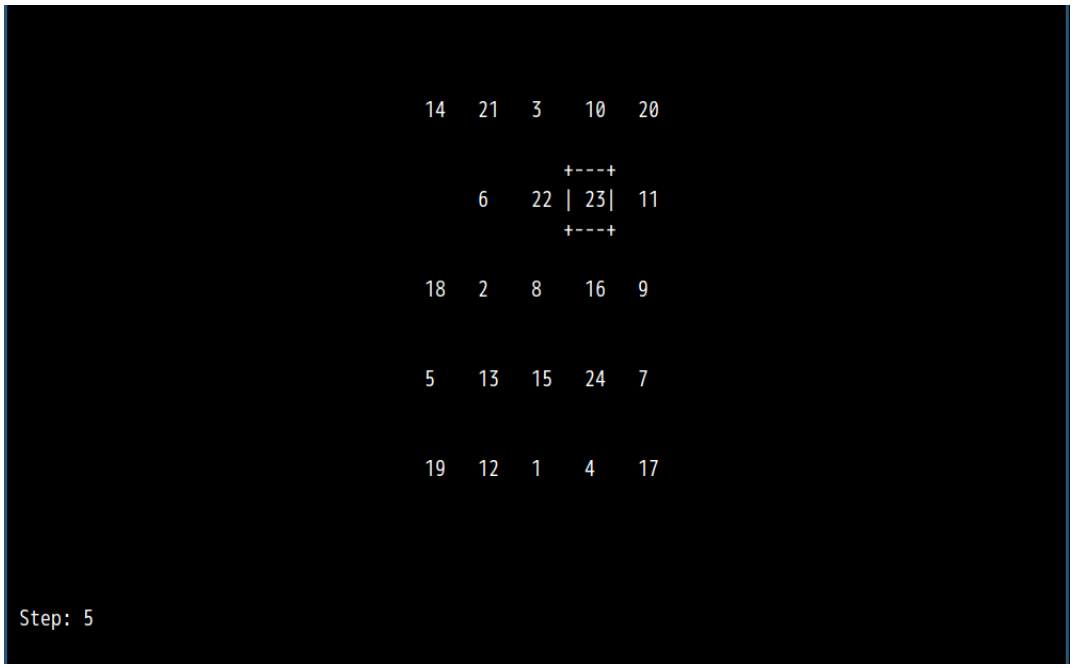

图 2: ncurses 界面示例

```
    int * loc; // -1: 空位

    int step;

    SelLocFunc slf;
    ConfStartFunc csf;
    NoteWinFunc nwf;
    NoteChangeFunc ncf;

};
```

其中, `nx`, `ny` 分别表示两个维度 x, y 的网格数 n_x, n_y. `eloc` 为正确图像的空格位置. `celoc` 为当前游戏状态的空格位置, `loc` 为一个数组, 保存每个实际位置的内容图块, 取值为内容图块的原始位置 (整数). `eloc`, `celoc`, `loc` 的取值按照公式

$$l = yn_x + x; \tag{1}$$

将二维坐标 `x`, `y` 转换为一维坐标 `l`. 其中 x, y 的取值范围分别为 $\{0, 1, \ldots, n_x - 1\}$ 及 $\{0, 1, \ldots, n_y - 1\}$. 例如, 图 2 中所示的状态下, 取 x 正方向向右, 取 y 正方向向下, 则 `loc`, `eloc`, `celoc` 的取值分别为

```
loc = {13, 20, 2, 9, 19, -1, 5, 21, 22, 10, 17, 1, 7, 15, 8, 4, 12,
   14, 23, 6, 18, 11, 0, 3, 16}
```

```
eloc = 24
celoc = 5
```

step 为用户已执行的操作步数. slf, csf, nwf, ncf 为 4 个回调函数接口, 用于与前端用户界面进行交互. 其函数接口定义为如下函数指针.

```
// \return: -1: 中止 0: 返回选择位置到 (x, y) 1: 返回位置到 dir: 0:
   x+ 1: x- 2: y+ 3: y-;
typedef int  (* SelLocFunc)  (SPDriver * sp, int * x, int * y, int *
    dir);
typedef int  (* ConfStartFunc)  (SPDriver * sp); // -1: 终止 0: 继续
/* x, y: 实际位置, cx, cy: 位于该位置的图块的原始坐标. cx == -1: 表
   示空位置.
 */
typedef void (* NoteChangeFunc) (SPDriver * sp, int x, int y, int cx
   , int cy);
typedef void (* NoteWinFunc) (SPDriver * sp, int res); // res = 0:
   成功, 1: 中止
```

首先, 游戏开始时, 引擎会调用 csf (ConfStartFunc) 函数询问用户是否开始游戏. 当每个位置所包含的图块的内容发生变化时, 引擎会调用 ncf(NoteChangeFunc) 通知前端. 其中, x, y 标记所发生变化的实际位置坐标, cx, cy 表示该位置所包含的内容图块的原始位置坐标. 例如, 如果对于图 2 所示状态, 引擎要通知前端最中心位置的状态, 会发起如下调用

```
sp->ncf (sp, 2, 2, 2, 1);
```

当引擎需要获取用户的滑动操作时, 会调用 slf (SelLocFunc) 函数. 用户可以通过函数的返回值表明三种不同类型的操作, 并将相应操作的参数返回到特定函数的指针参数中. 具体含义为:

- 函数返回 -1 表示强制中止游戏.
- 函数返回 0 表示用户选择了一个与当前空格位于同行或同列的位置做一次行或列的滑动. 用户所选择的位置需要被返回到指针参数 x, y 中. 如果用户所选择的位置与当前空格不在同行或同列中, 则引擎会重新调用本函数让用户重新选择.
- 函数返回 1 表示用户选择了一个移动方向, 以完成当前空格与相邻一个空格的置换. 移动方向需要被返回到指针参数 dir 中, 取值为 $\{0, 1, 2, 3\}$, 具体含义为:

0 当前空格的 x 坐标减少.

1 当前空格的 x 坐标增加.

2 当前空格的 y 坐标减少.

3 当前空格的 y 坐标增加.

当用户完成拼图或游戏被用户强制中止时, 引擎会调用函数 nwf(NoteWinFunc) 通知前端. res 为 0 表示用户成功完成拼图, 为 1 表示用户强制中止游戏.

4 引擎使用流程

首先, 前端的开发者需要定义好第 3 节所述的 SPDriver 结构中的 4 个回调函数 (slf, csf, nwf, ncf). 然后可按如下代码启动游戏和释放引擎.

```
SPDriver * sp = InitSP (nx, ny, ex, ey, slf, csf, nwf, ncf);
RunSP (sp);
FreeSP (sp);
```

其中, 各函数的定义为

```
SPDriver * InitSP (int nx, int ny, int ex, int ey, SelLocFunc slf,
   ConfStartFunc csf, NoteWinFunc nwf, NoteChangeFunc ncf);
void RunSP (SPDriver * sp);
void FreeSP (SPDriver * sp);
```

其中, nx, ny 为两个维度的图块个数. ex, ey 为原始的空格坐标. slf, csf, nwf, ncf 为上述 4 个接口回调函数. 另外, 还有如下几个辅助函数可获取游戏状态.

```
void GetLoc (SPDriver * sp, int x, int y, int * ox, int * oy);
```

用于获取位于位置坐标 x, y 处的图块的原始位置坐标, 结果返回到 ox, oy 中. 返回 -1 表示该位置当前为空位.

```
int IsWin (SPDriver * sp); // 1: 胜利, 0: 未胜利
```

用于判断用户是否已经完成拼图, 返回 1 表示已完成, 0 表示未完成.