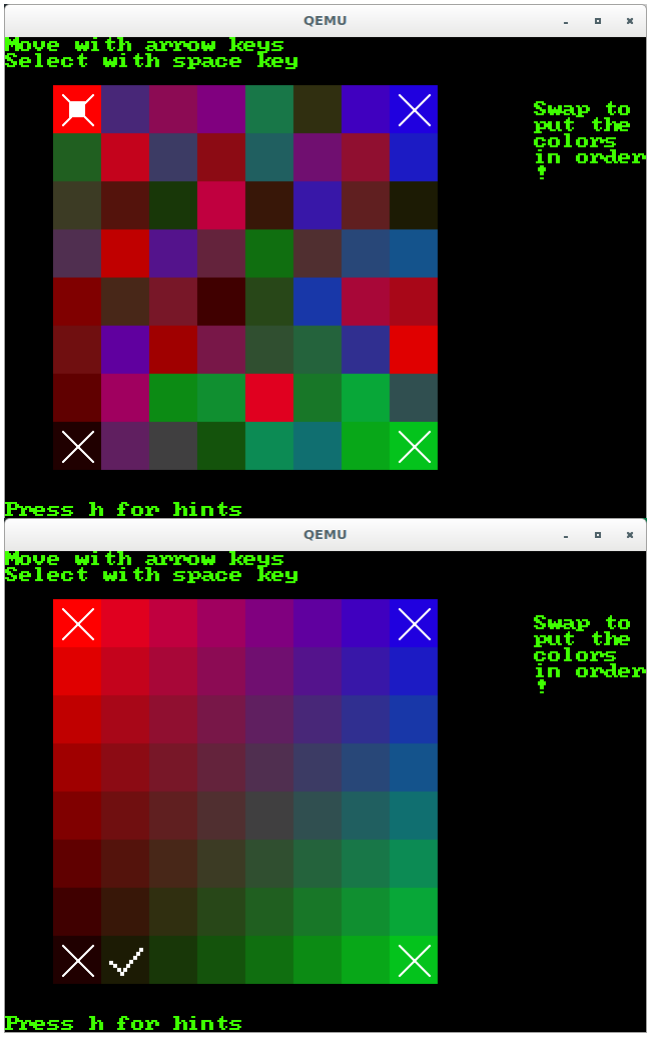
L0 直接运行在硬件上的小游戏(amgame)

游戏截图



游戏内容

游戏会随机*打乱GRID×GRID（在include/game.h中定义）个方块。其中，颜色为渐变色*。
玩家通过上下左右键移动，按下空格选中方块。选中两个方块后会自动交换。玩家要使界面内的颜色重新恢复原本的渐变色。

注1：随机指的是游戏文件中调用了srand(uptime())，基于rand函数的特性及uptime自身的不确定性，可以近似认为游戏达到了随机
注2：渐变色指，在初始的图中，同一行/列的方格，RGB值为三个等差数列

文件结构

```
src
├── draw.c
├── font.c
├── game.c
├── logic.c
└── shape.c
```

我的源代码分为5个文件。
game.c为游戏主循环，只包含主要的函数调用。
shape.c提供绘制基本图形的函数，如画叉、圆、箭头、打钩、光标及填充指定方格等。
font.c提供在图形界面上打印字符的函数，包含ASCII编码下所有字符的点阵，一个打印单个字符的函数（都是静态），打印字符串的函数（非静态）
draw.c提供较复杂的绘图的接口，包括初始化屏幕和在光标选中格打印光标或提示（由print_flag决定）
logic.c提供复杂逻辑的判定，包括游戏初始化、交换两个方格、计算颜色渐变值、获得下一个输入按键、对输入按键进行处理。

设计亮点

- 考虑到如果在主循环中进行判定while(uptime()< next_frame)，会导致按键的延迟，因此，我没有在主循环中使用时间相关的判定。只在少数需要体现出时间特性的部分单独加入了判断（如光标闪烁频率、长按按键的反馈）。这样可以做到，玩家按下按键后，第一时间*处理该按键信息，且长按时表现出若干毫秒处理一次的特性。
- 考虑到native和qemu的分辨率不同（同时还要考虑兼容其它分辨率），include/game.h 中 SIDE 被定义成 16会导致在不同分辨率下的效果差很多。所以我经过计算和尝试，将其改成w/40。使代码能够适应大多数分辨率。同理，draw_str中提供的size参数，我也使用了SIDE以实现自动适应屏幕的效果。但由于整型数精度问题，文字的适应效果有限。
- 考虑到游戏难度较大（作者自己都玩不过），我增加了Hint功能，按下h即可显示提示
- 为了节约空间，没有使用canvas数组，因此在打印时，为了表现出多层的效果，在进行判定时代码量稍多一点。
- 由于裸机本身不提供透明通道，alpha参数被我用来记录其它数值（该方格是否为固定方格）