基于 μC/OS-Ⅱ和 μCGUI 的推箱子游戏
实现说明

2014 年 11 月

121250101 缪晓伟

目　录

一、概述

本次作业基于 μC/OS-Ⅱ和 μCGUI 进行构建，在事先完成了对开发板配置和初始化的模版的基础上，构建了一款最为简单的、仅有一个关卡的推箱子游戏。

二、实现说明

1.中断响应机制

在实现此游戏的代码中，依然使用了全局变量 state 记录游戏过程中的当前状态。在外部中断发生，即用户按下了开发板上四个按钮的情况下，通过表驱动的方式，在 O(1)的时间复杂度下完成在每个状态按下不同按键对应功能的执行。

以下为按键中断服务子程序中的核心代码。

```
if(state<=2)
{
    if(key % 2 == 1 && key<=7)
        stateTable[state][(key-1)/2]();
    if(rEINTPEND & 1<<11)
        rEINTPEND |= 1<<11;
    if(rEINTPEND & 1<<19)
        rEINTPEND |= 1<<19;
}
```

代码中所示的 stateTable 在代码运行之初进行设定。

```
//函数指针表初始化，分别表示不同状态下按下不同按键调用的函数
stateTable[0][0]=stateTable[0][1]=stateTable[0][2]=stat
eTable[0][3]=enterGame;
stateTable[1][0]=pushDown;
stateTable[1][1]=pushLeft;
stateTable[1][2]=pushRight;
stateTable[1][3]=pushUp;
stateTable[2][0]=stateTable[2][1]=stateTable[2][2]=stat
eTable[2][3]=drawInitScreen;
```

2.重要游戏逻辑

1) 地图的存储

推箱子游戏中，地图上的基本元素有五个：围墙、地面、箱子、箱子目标点和人物。因此，在游戏中，使用 1 个字节大小的数据单元即可完成对所有地图上可能情形的记录。同时，由于一些组合情况并不可能出现在地图上，这里使用宏预先定义了可能存在的状态常量。

```
//格子状态预置
#define GROUND 0x1
#define SPOT 0x2
#define WALL 0x4
#define BOX 0x8
#define CHARACTER 0x10
#define BOX_INPLACE (BOX|SPOT)
#define BOX_ONGROUND (BOX|GROUND)
#define CHARACTER_ON_SPOT (CHARACTER|SPOT)
#define CHARACTER_ON_GROUND (CHARACTER|GROUND)
#define OBSTCALE (WALL|BOX)
```

有了上述定义，初始地图的设置就变得很容易了，直接对一个二维数组进行初始化即可。

```
const U8 originalMap[8][8]={
    {GROUND,GROUND,WALL,WALL,WALL,GROUND,GROUND,GROUND},
    {GROUND,GROUND,WALL,SPOT,WALL,GROUND,GROUND,GROUND},
    {GROUND,GROUND,WALL,GROUND,WALL,WALL,WALL,WALL},
    {WALL,WALL,WALL,BOX_ONGROUND,GROUND,BOX_ONGROUND,SPOT,W
ALL},
    {WALL,SPOT,GROUND,BOX_ONGROUND,CHARACTER_ON_GROUND,WALL
,WALL,WALL},
    {WALL,WALL,WALL,WALL,BOX_ONGROUND,WALL,GROUND,GROUND},
    {GROUND,GROUND,GROUND,WALL,SPOT,WALL,GROUND,GROUND},
    {GROUND,GROUND,GROUND,WALL,WALL,WALL,GROUND,GROUND}
};
```

游戏实际执行过程中，由于地图状态会随时发生改变，因此需要第二个二维数组记录游戏的当前格局，以免影响地图的完整性。

2) 动作合法性的判断

根据推箱子的游戏规则，游戏中的人物可以在不推箱子的情况下随意走动，也可以推动一个箱子一格距离远，并且人物向相同的方向移动。如果没有上节所述宏的定义，动作合法性的判断会产生很大的困难；但有了宏的定义，原本复杂的状态判断就可以通过位运算直接解决了。

```
BOOLEAN   canPush(int   xPos,int   yPos,int   xDirection,int
yDirection)
{
    int newX=xPos+xDirection;
    int newY=yPos+yDirection;
    //地图外不能推
    if(newX>=0&&newX<=7&&newY>=0&&newY<=7)
    {
        //空地（没有障碍物）可以推
        if(!(currentMap[newX][newY]&OBSTCALE))
            return TRUE;
        //墙不能推
        if(currentMap[newX][newY]&WALL)
            return FALSE;
        //单个箱子可以推
        if(currentMap[newX][newY]&BOX)
        {
            //May be dangerous, but map can guarantee index
won't overflow...
    if(!(currentMap[newX+xDirection][newY+yDirection]&OBSTC
ALE))
                return TRUE;
            else
                return FALSE;
        }
        return TRUE;
    }
    else
        return FALSE;
}
```

代码中下划线部分的语句可能会产生数组越界的风险。但考虑到地图四周都会有围墙阻挡，并不会出现将边缘的箱子推“出”地图的情况，因此此处没有做更多的安全检查。只要保证地图在建立时没有问题，这里的代码就不会产生问题。

3) 游戏地图的更新

由于游戏能够推动的物体只有箱子，实际更新地图的代码就非常简单。如果有箱子要推的话，把箱子"推"到下一个位置，同时将人物移动至原有箱子的位置即可。地图上的不同状态已经通过宏定义好，因此直接通过位运算置位即可。

```
//前方有箱子，推出去的也只能是箱子
if(currentMap[newX][newY]&BOX)
{
    currentMap[newerX][newerY]|=BOX;
    currentMap[newX][newY]&=~BOX;
    currentMap[newX][newY]|=CHARACTER;
    currentMap[xPos][yPos]&=~CHARACTER;
    drawBlockAt(newerX,newerY);
    drawBlockAt(newX,newY);
    drawBlockAt(xPos,yPos);
}
else
{
    currentMap[newX][newY]|=CHARACTER;
    currentMap[xPos][yPos]&=~CHARACTER;
    drawBlockAt(newX,newY);
    drawBlockAt(xPos,yPos);
}
```

3.游戏图案的绘制

游戏中可能出现的图案，事先在 PC 机上处理好后，通过 μC-GUI-BitmapConvert 工具处理成 μCGUI 系统能够识别的结构体后，已经硬编码在头文件 pic.h 中。实际绘制时，通过辨别某个格子的状态，直接读取相应的位图结构，调用 API 完成绘制即可。

```
void drawSingleBitmap(U8 state,int line,int coloum)
{
    int startX=60+30*coloum;
    int startY=60+30*line;
    const GUI_BITMAP *pic=NULL;
    switch(state)
    {
        case WALL:
            pic=&bmwall;
            break;
```

```
            case GROUND:
                pic=&bmground;
                break;
            case SPOT:
                pic=&bmspot;
                break;
            case BOX_ONGROUND:
                pic=&bmbox;
                break;
            case BOX_INPLACE:
                pic=&bmbox_inplace;
                break;
            case CHARACTER_ON_SPOT:
            case CHARACTER_ON_GROUND:
                pic=&bmcharacter;
                break;
        }
        GUI_DrawBitmap(pic,startX,startY);
    }
```

三、实际效果

由于设备所限，无法在 PC 机上完全模拟游戏的运行，因此这里无法提供截图。实际的运行效果可以通过将 bin 文件烧入开发板中实际运行的方式查看。