基于 Easyx 设计的推箱子

一、需求分析

1、 基本游戏逻辑

2、 图形化界面

3、 随机生成简单地图

4、 加上主菜单界面，按键，文字显示等功能

二、设计和测试

推箱子作为一项周所周知的游戏，十分简单。玩家主要目标就是将地图上所有的箱子在越过墙壁等障碍下通过上下左右键推到指定位置，以达到胜利目标。

基本游戏逻辑：

这个游戏的底层逻辑也很清楚，地图上无非就分为 6 种方块，分别是：路、墙壁、人物、箱子、指定位置、推到指定位置的箱子。每次按下按键前先扫描人物所在位置，然后对上下左右进行分类。以向上为例，如果人物的上一格为路，那么将人物的上一格变为人物，人物这一格变为路。如果上一格为箱子指定点，则上一格变为人，人物这一格变为路。如果上一格为箱子，就要再检测箱子的上一格，如果是路就都按前两个做法将人物和箱子向上挪一个。如果是箱子指定点，就将人物与箱子都向上挪一格，并将箱子变换一个形状。

```
void keyDown()
{
    int userKey = _getch(); //不可见输入
    //定位：找到人的位置
    int i = 0;
    int j = 0;
    for (i = 1; i < ROW; i++)
    {
        for (j = 1; j < COL; j++)
        {
            if (map[cas][i][j] == 5 || map[cas][i][j] == 8)
            {
                goto NEXT;
            }
        }
    }
NEXT:
```

```
case 'W':
case 'w':
case 72:
    if (map[cas][i - 1][j] == 0 || map[cas][i - 1][j] == 3)
    {
        map[cas][i][j] -= 5;
        map[cas][i - 1][j] += 5;
        _count++;
    }
    if (map[cas][i - 1][j] == 4 || map[cas][i - 1][j] == 7)
    {
        if (map[cas][i - 2][j] == 0 || map[cas][i - 2][j] == 3)
        {
            map[cas][i][j] -= 5;
            map[cas][i - 1][j] += 1;
            map[cas][i - 2][j] += 4;
            _count++;
        }
    }
    break;
```

判断游戏结束标志:
地图上没有未到达指定点的箱子。

```
int gameOver()
{
    //地图上没有箱子就可以结束
    for (int i = 0; i < ROW; i++)
    {
        for (int j = 0; j < COL; j++)
        {
            if (map[cas][i][j] == 4)
            {
                return 0;
            }
        }
    }
    return 1;
}
```

图形化界面：

其中 0 为路，1 为墙，3 为箱子指定点，4 为箱子，5 或 8 为人物，7 为箱子进入指定点。

```
void  drawGraph()
{
    for (int i = 0; i < ROW; i++)
    {
        for (int j = 0; j < COL; j++)
        {
            //算贴图的坐标
            int x = 50 * j;
            int y = 50 * i;
            switch (map[cas][i][j])
            {
            case 0:
                //一个汉字符号占用两个位置
                //printf("  ");
                putimage(x, y, img + 0);
                break;
            case 1:
                putimage(x, y, img + 1);
                //printf("■");
                break;
            case 3:
                putimage(x, y, img + 2);
                //printf("☆");
                break;
            case 4:
                putimage(x, y, img + 3);
                //printf("★");
                break;
            case 5:
            case 8:
                putimage(x, y, img + 4);
                //printf("人");
                break;
            case 7:
                putimage(x, y, img + 5);
                //printf("●");
                break;
            }
        }
```

在主函数中调用时，每次按下按盘时，进行图像的刷新，实现图像的更新。

```
void  drawGraph1()//绘制背景图
{
    for (int i = 0; i < ROW; i++)
    {
        for (int j = 0; j < COL; j++)
        {
            int x = 50 * j;
            int y = 50 * i;
            putimage(x, y, bacground);
        }
    }
}
```

随机生成简单地图：

看似随机生成地图简单，但是要考虑一件事情：那就是这个地图是否有解。也就是是否真的能将所有的箱子推进目标点。

通过建立 Mymap 类，封装一系列函数，用于生成随机地图。这段有点复杂，就不一一阐述了。详情参考 cpp 文件。

```
class MyMap {
public:
    int Map[_SIZE][_SIZE]{ WALL };
private:
    int X1{}, X2{}, Y1{}, Y2{};//分别为1、2步之前的坐标
    int X{}, Y{}, Direction{};//当前坐标、方向
    int PX{}, PY{};//玩家坐标
    int StartX{}, StartY{};//记录起点，用于生成箱子

    bool test() {//一定程度避免随机数整烂活;
        int test{};
        for (int i = 0; i < _SIZE; i++) {
            for (int j = 0; j < _SIZE; j++) {
                if (Map[i][j] == 3)test += 1;
                if (Map[i][j] == 4)test += 10;
                if (Map[i][j] == 5)test += 100;
            }
        }
        return (test == 100 + 11 * DUISHU);
    }
```

加入屏幕上的按键以及文字显示：

文字显示较为简单，直接调用函数设定目标文字的坐标以及显示内容即可。

而按键较为复杂，需要通过封装类 Button 实现。详情间见 cpp 文件。

```
class Button
{
private:
    int x; // 按钮左上角x坐标
    int y; // 按钮左上角y坐标
    int width; // 按钮宽度
    int height; // 按钮高度
    float scale; // 缩放比例，用于实现鼠标悬停效果
    bool isMouseOver; // 表示鼠标是否在按钮上方
    bool has_been_clicked;//表示此按钮是否被点击过（主要用于难度选择）
    wstring text; // 按钮文本

public:

    Button(int x, int y, int width, int height, const wstring& text)
        : x(x), y(y), width(width), height(height), text(text), scale(1.0f), isMouseOver(false),has_beer
    {
    }

    // 检查鼠标是否在按钮上方
    void checkMouseOver(int mouseX, int mouseY)
    {
        isMouseOver = (mouseX >= x && mouseX <= x + width && mouseY >= y && mouseY <= y + height);
```

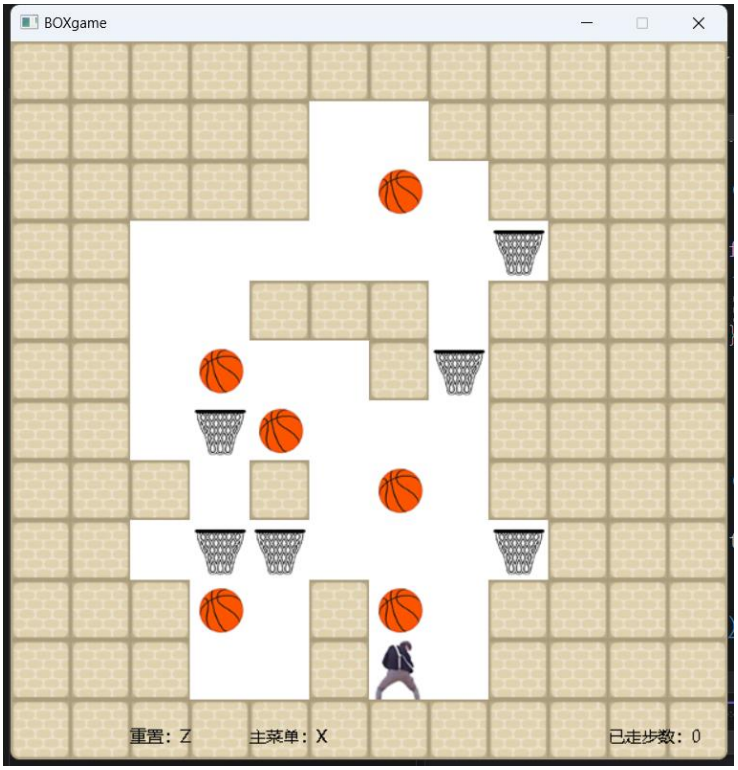
```
outtextxy(500, 570, L"已走步数：");
outtextxy(100, 570, L"重置：Z");
outtextxy(200, 570, L"主菜单：X");
```

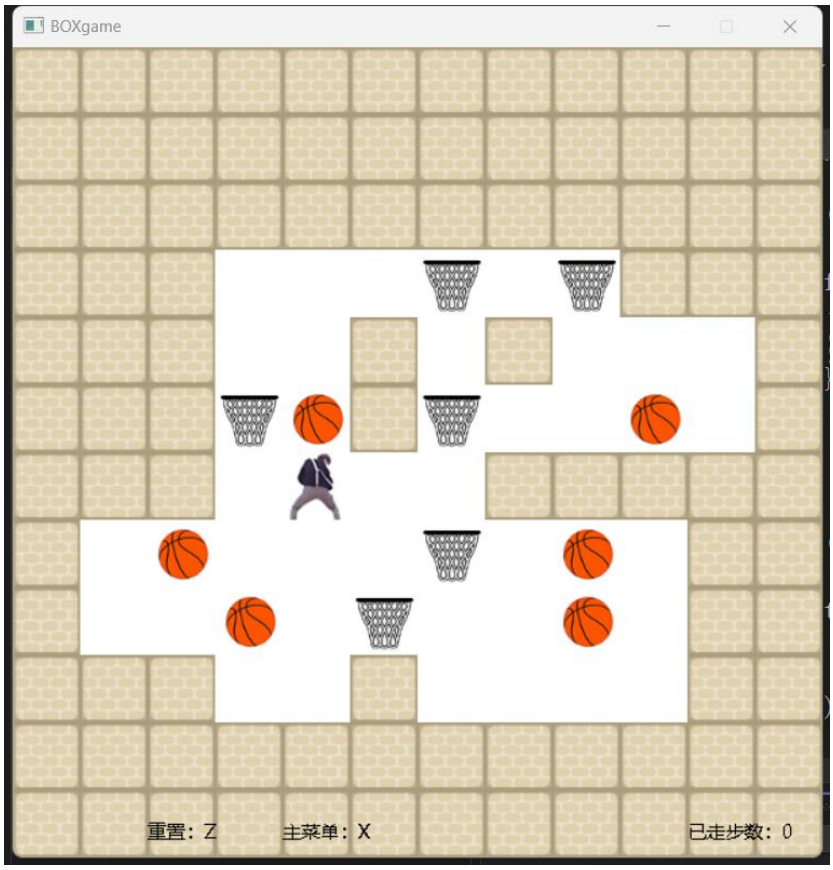
整体游戏规划：

在 main 函数中先设计一个主菜单界面，通过等待玩家选中游戏难度，进入指定难度的关卡，难度等级分别为：简单，普通，困难。

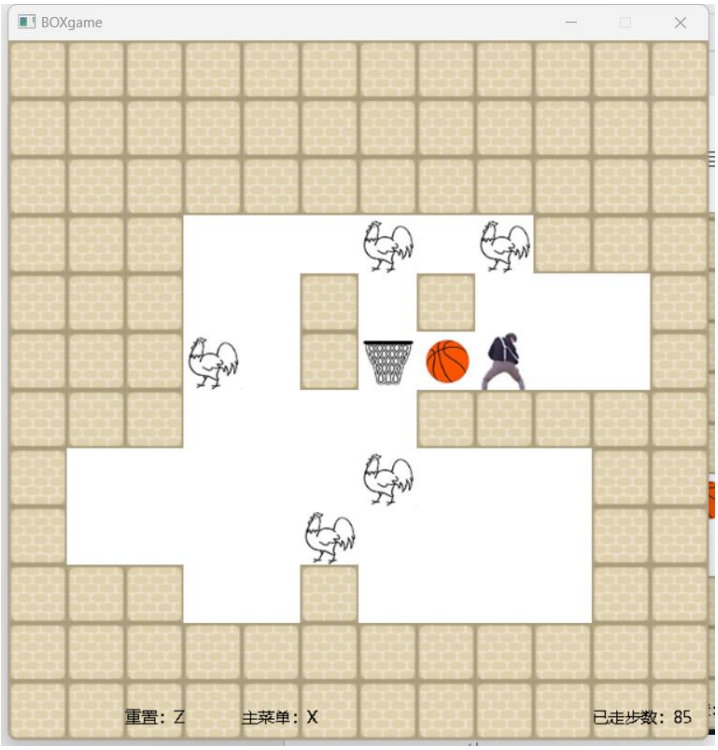
通过选择难度等级，进入游戏，每个难度等级都有 4 关，可重复进入。

游戏测试：





注意到这里我们用的人物是我的偶像坤坤，箱子也被替换为专属道具篮球。



右下角有已走步数提醒，重置可以按 Z 键，返回主菜单可以按 X 键。