北京理工大學

本科生大作业设计（论文）

咸鱼寻宝记

学　　院：计算机学院

专　　业：计算机科学与技术

学生姓名：祁瑜，潘宣文，龚致楹，张居位

学　　号：1120203282（龚）,1120203302(张)，1120202558（祁），1120202720（潘）

指导教师：张华平

2022 年　12 月　31 日

原创性声明

本人郑重声明：所呈交的毕业设计（论文），是本人在指导老师的指导下独立进行研究所取得的成果。除文中已经注明引用的内容外，本文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

特此申明。

本人签名：　　日 期：　　年　月　日

关于使用授权的声明

本人完全了解北京理工大学有关保管、使用毕业设计（论文）的规定，其中包括：①学校有权保管、并向有关部门送交本毕业设计（论文）的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存本毕业设计（论文）；③学校可允许本毕业设计（论文）被查阅或借阅；④学校可以学术交流为目的，复制赠送和交换本毕业设计（论文）；⑤学校可以公布本毕业设计（论文）的全部或部分内容。

本人签名：　　日 期：　　年　月　日

指导老师签名：　　日 期：　　年　月　日

咸鱼寻宝记

摘　要

推箱子这款游戏最初名为《仓库番》，由今林宏行先生于 1981 年春天使用 BASIC 语言制作，因为其简单的规则以及复杂多变的地图玩法在今天仍有一大批粉丝。本游戏的灵感也来源于此，我们使用基本的 win32 汇编语言编写的这款《咸鱼寻宝记》游戏，在基本的推箱子游戏上加入了一些新的功能，并且原创了很多张新地图。我们希望在保留原游戏的玩法上，给游戏增添更多趣味。游戏以 32 位汇编窗口编程基础的消息机制为主体框架，自定义了功能函数以满足游戏的功能需要。

关键词：推箱子；win32 汇编

第 1 章 背景与需求说明

1.1 游戏理念的提出

在当今 3A 游戏横行霸道的背景下，我们小组希望返璞归真，打造玩法最初始，操作最便捷，同时最能引起大家共鸣的“全民游戏”因此我们选择了《推箱子》这个经典游戏 ip 作为基础，以此打造我们全新的创意游戏《咸鱼寻宝记》。

1.2 游戏背景

你是一条失去理想的咸鱼，突然有一天你竟然穿越到了异世界！一个声音告诉你只要将所有的箱子推到他们本应的位置上，你就可以获得无上的财宝。这看似会是西西弗斯推石头一样的工作，对你来说真的如此吗？

1.3 需求说明

我们需要使用 win32 汇编语言进行游戏编写。

本游戏需要实现三个界面，分别为初始化界面，选择关卡界面与游戏界面。

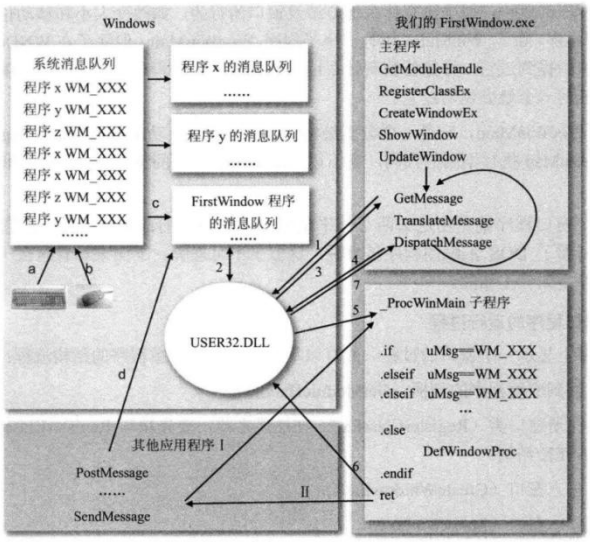
初始化界面		
元素名称	类型	功能
游戏名以及主创人员的信息	位图 bitmap	展示主要信息
“开始游戏”按钮	按钮 IButtonEditor	点击即进入选择关卡界面
选择关卡界面		
10 个关卡选择按钮	按钮 IButtonEditor	点击即进入对应的关卡游戏界面
游戏界面		
“重新开始”按钮	按钮 IButtonEditor	点击即重新开始本局游戏
“选择关卡”按钮	按钮 IButtonEditor	点击即进入选择关卡界面，

		重新选关
“步数”文字	文本 IStatEditor	记录本关游戏步数
“关卡”文字	文本 IStatEditor	标识关卡数
“剩余穿箱次数”文字	文本 IStatEditor	标识剩余技能使用次数

除了上述控件之外，我们还需编写有关的控制逻辑，详细内容在下面几章给出，本章不再赘述。

第 2 章 总体功能框架设计

本游戏的控制逻辑采用经典的 win32 汇编中的窗口编程规范，使用消息队列来传输各个控件之间的逻辑控制流：



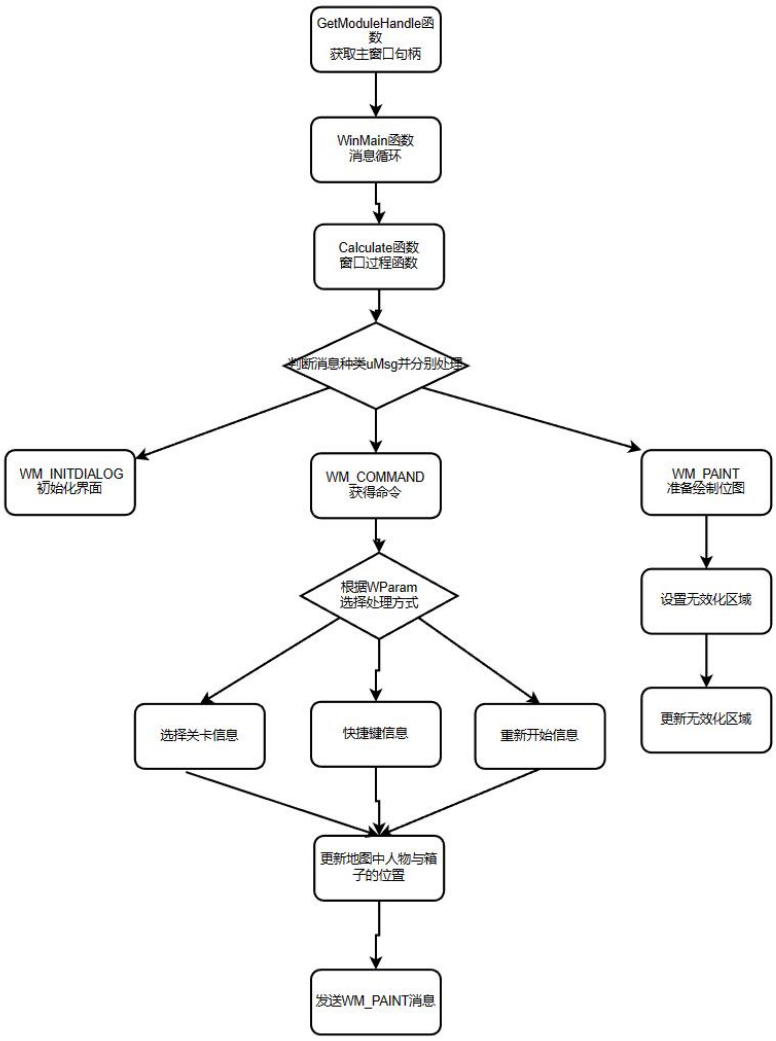
程序从我们自己编写的 WinMain 函数开始，初始化窗口，并且建立消息循环机制。

我们编写 CreateLevelMap 函数来创建当前关卡地图，其底层调用 syDrawMap 函数进行地图的绘制，我们所有的图像均以位图的形式绘制。

在消息循环中，我们编写自己的消息处理函数 Calculate 来处理各种消息。

此外，我们调用键盘上的上下左右键作为游戏时人物移动的按键，每一种按键都对应着一种消息，需要被消息处理函数接受并处理，才能重新绘制人物的位置。

我们编写上下左右函数，以及特殊技能函数，来处理摁下对应键后程序应该对地图作出何种改变，接下来交由绘制函数重新绘制位图。


GetModuleHandle函数
获取主窗口句柄
WinMain函数
消息循环
Calculate函数
窗口过程函数
判断消息种类uMsg并分别处理
WM_INITDIALOG
初始化界面
WM_COMMAND
获得命令
WM_PAINT
准备绘制位图
根据WParam
选择处理方式
设置无效化区域
选择关卡信息
快捷键信息
重新开始信息
更新无效化区域
更新地图中人物与箱子的位置
发送WM_PAINT消息

第 3 章 详细方案设计

3.1 地图各项的底层实现

本游戏采取传统的静态地图，每关的地图固定且全部可见。每张地图有五个组成部分，如图 3-1 所示：1.一只可以上下左右移动的咸鱼；2.若干个初始时即存在场上的木箱，可被移动；3.石墙，是咸鱼移动的边界；4.若干个目标点，通关需要把所有木箱移动到目标点；5.供咸鱼自由移动的普通单位

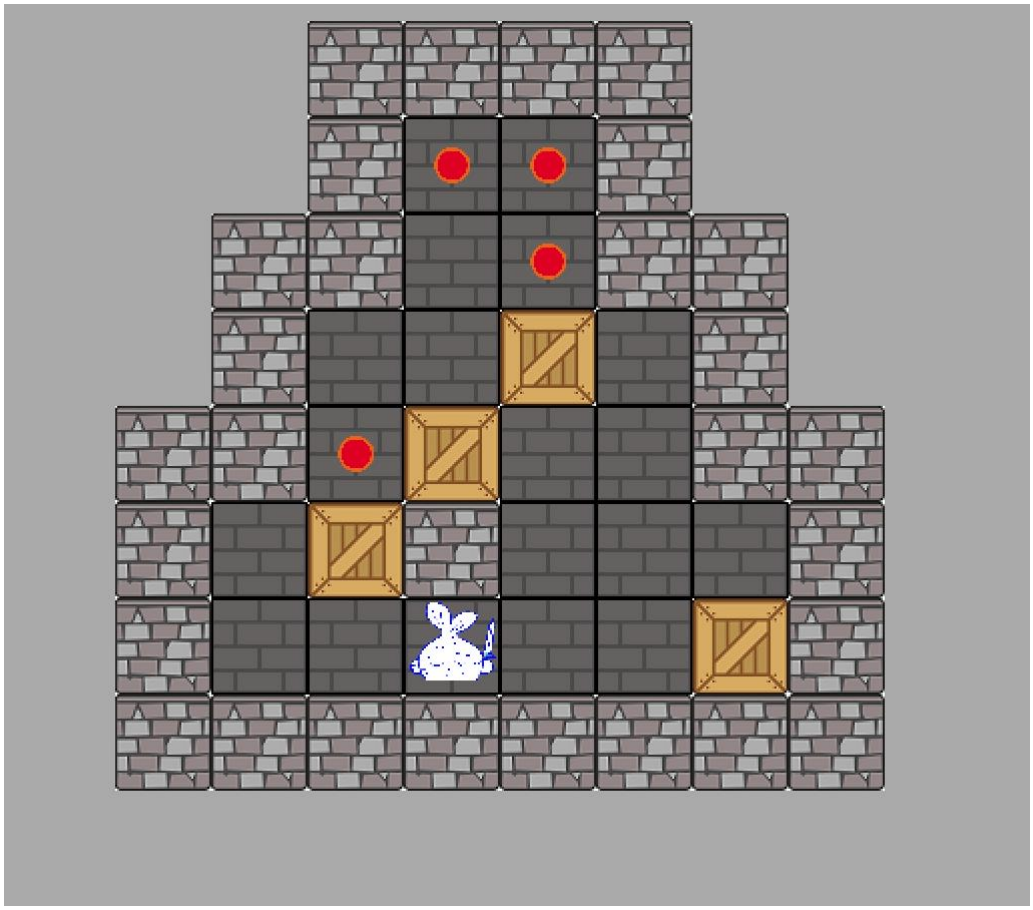


图 3-1 游戏示意图

在初始化地图时，我们使用 CurrentMapText 数组来记录地图中每个单位对应的属性，作为背景的单位记为 0；作为石墙的单位记为 1；普通单位记为 2；咸鱼所在的单位记为 3；木箱所在的单位记为 4；目标点所在的单位记为 5。通过不断更新各单位的属性，实现咸鱼与木箱的移动。

3.2 游戏机制

本游戏保持了推箱子的原有机制：可以通过上、下、左、右方向键控制咸鱼的移动；咸鱼只能移动至普通方格或目标点；若木箱与咸鱼相邻，则可以使咸鱼推动木箱共同前进一个单位。在此基础上，我们增加了拉木箱的动作，若木箱与咸鱼相邻，则可以按住 crtl+方向键使咸鱼拉动木箱共同前进一个单位。我们还增加了特殊的跳跃技能。若木箱与咸鱼相邻，且木箱的前方是可进入区域（普通方格或目标点）则按住 shift+方向键，咸鱼可以穿过木箱，到达木箱的前方相邻单位。此技能每关只可使用 5 次

第 4 章 关键算法与代码

4.1 普通移动兼推动木箱 MoveUp/MoveDown/MoveLeft/MoveRight

咸鱼向上移动或向上推动木箱的代码如 4-1 所示：利用 CurrPosition 获取咸鱼的当前位置，判断此位置上方的相邻单位是否可进入或是否为木箱。如果不可进入或与咸鱼相邻的木箱的上方不可进入，步数加一，咸鱼仅朝向上方而不移动；否则步数加一，咸鱼朝向上且向上移动一个单位，更改相应单位的属性，刷新地图。下、左、右同理。

```
MoveUp proc
    ; 找到当前位置
    xor esi, esi
    mov esi, CurrPosition
    mov edi, esi
    sub edi, 10; edi记录上方位置

    ; 设置角色脸朝向
    invoke sySetPlayerFace, SY_FACE_UP

    ; 判断上方单位属性
    ; 如果是空地或结束点，移动
    .if CurrentMapText[edi * 4] == 2 || CurrentMapText[edi * 4] == 5
        mov  CurrPosition, edi; 改变咸鱼的当前位置
        mov dword ptr CurrentMapText[edi * 4], 3; 改变上方单位属性
        mov eax, OriginMapText[esi * 4]
        mov CurrentMapText[esi * 4], eax
        ; 刷新格子
        invoke syUpdateGrid, edi
        invoke syUpdateGrid, esi

    ; 如果是箱子
    .elseif CurrentMapText[edi * 4] == 4
        ; 判断箱子那边是什么
        xor ecx, ecx
        mov ecx, edi
        sub ecx, 10; ecx是咸鱼的上上方位置

        ; 如果是围墙或箱子
```

```
        .if CurrentMapText[ecx * 4] == 1 || CurrentMapText[ecx * 4] == 4
            ; 刷新格子
            invoke syUpdateGrid, esi
        .else
            ; 只可能是空地或存放点，可以移动
            mov CurrPosition, edi; 改变咸鱼的当前位置
            mov dword ptr CurrentMapText[ecx * 4], 4
            mov dword ptr CurrentMapText[edi * 4], 3
            mov eax, OriginMapText[esi * 4]
            mov CurrentMapText[esi * 4], eax
            ; 刷新格子
            invoke syUpdateGrid, ecx
            invoke syUpdateGrid, edi
            invoke syUpdateGrid, esi

        .endif

    .else
        ; 是墙
        invoke syUpdateGrid, esi
    .endif
    ret

MoveUp endp
```

4-1

4.2 拉动木箱 DragUp/DragDown/DragLeft/DragRight

咸鱼向上拉动木箱的代码如 4-2 所示：利用 CurrPosition 获取咸鱼当前位置，若咸鱼的下方相邻单位不是木箱，则此次移动视为普通的上移，与 4-1 完全相同；若咸鱼的下方相邻单位是木箱且咸鱼上方可以进入，则咸鱼朝向上，拉动木箱共同向上一个单位，步数加一；若咸鱼的下方相邻单位是木箱且咸鱼上方不可进入，则仅使咸鱼朝向上方，步数加一。下、左、右同理。

```
DragUp proc
    xor esi, esi
    mov esi, CurrPosition
    mov edi, esi
    sub edi, 10
    mov ebx, esi
```

```
add ebx, 10

invoke sySetPlayerFace, SY_FACE_UP

.if CurrentMapText[edi * 4] == 1 || CurrentMapText[edi * 4] == 4
    invoke syUpdateGrid, esi

.else
    mov CurrPosition, edi
    .if CurrentMapText[ebx * 4] == 4
        mov dword ptr CurrentMapText[edi * 4], 3
        mov dword ptr CurrentMapText[esi * 4], 4
        mov eax, OriginMapText[ebx * 4]
        mov CurrentMapText[ebx * 4], eax
        invoke syUpdateGrid, ebx
        invoke syUpdateGrid, esi
        invoke syUpdateGrid, edi

    .else
        mov dword ptr CurrentMapText[edi * 4], 3
        mov eax, OriginMapText[esi * 4]
        mov dword ptr CurrentMapText[esi * 4], eax

        invoke syUpdateGrid, edi
        invoke syUpdateGrid, esi
    .endif
.endif

ret

DragUp endp
```

4-2

4.3 跳过木箱 JumpUp/JumpDown/JumpLeft/JumpRight

咸鱼向上跳跃的代码如 4-3 所示：利用 CurrPosition 获取咸鱼的当前位置，若咸鱼上方不是木箱且不可进入，则步数加一，咸鱼朝向上而不移动，剩余可穿箱次数减一；若咸鱼上方不是木箱且可以进入，则视作普通的向上移动，步数加一，剩余可穿箱次数减一；若咸鱼上方是木箱且木箱上方不可进入，则咸鱼朝向上而不移动，剩余可穿箱次数减一；若咸鱼上方是木箱且木箱上方可进入，则咸鱼移动到木箱上

方。下、左、右同理。

```
JumpUp proc
    ; 找到当前人的位置
    xor esi, esi
    mov esi, CurrPosition; 假设CurrPosition记录当前人的位置, esi记录当前人位置
    mov edi, esi
    sub edi, 10; edi记录人的上方位置

    ; 设置角色脸朝向
    invoke sySetPlayerFace, SY_FACE_UP

    ; 判断上方格子类型
    ; 如果是空地或结束点, 人移动
    .if CurrentMapText[edi * 4] == 2 || CurrentMapText[edi * 4] == 5
        mov  CurrPosition, edi; 改变人的当前位置
        mov dword ptr CurrentMapText[edi * 4], 3; 改变上方方格属性
        mov eax, OriginMapText[esi * 4]
        mov CurrentMapText[esi * 4], eax
        ; 刷新格子
        invoke syUpdateGrid, edi
        invoke syUpdateGrid, esi


    .elseif CurrentMapText[edi * 4] == 4
        ; 判断箱子那边是什么
        xor ecx, ecx
        mov ecx, edi
        sub ecx, 10; ecx是人的上上方位置

        ; 如果是围墙或箱子
        .if CurrentMapText[ecx * 4] == 1 || CurrentMapText[ecx * 4] == 4
            ; 刷新格子
            invoke syUpdateGrid, esi

        .elseif currentBackStep > 0
            ; 只可能是空地或存放点，可以移动
            mov CurrPosition, ecx; 改变人的当前位置
            mov dword ptr CurrentMapText[ecx * 4], 3
            mov eax, OriginMapText[esi * 4]
            mov CurrentMapText[esi * 4], eax
            ; 刷新格子
```

```
            invoke syUpdateGrid, ecx
            invoke syUpdateGrid, esi

        .elseif currentBackStep <= 0
            invoke syUpdateGrid, esi

        .endif

    .else
        ; 是墙
        invoke syUpdateGrid, esi
    .endif
    ret
JumpUp endp
```

4-3

4.4 判断是否胜利 JudgeWin

判断本关是否胜利的代码如4-4所示，每当咸鱼移动、地图刷新时，调用JudgeWin函数判断本关是否胜利结束。若所有初始时为目标点的单位全部变为木箱，即所有OriginMapText值为5的单位的CurrentMapText值全部变为4，则本关胜利结束；否则继续游戏。胜利后自动跳转下一关。

```
; 判断输赢
JudgeWin proc
    ; 若图中不出现属性5，证明箱子全部到位
    xor eax, eax
    xor ebx, ebx; ebx记录图中箱子存放点数量
    mov eax, 0
    .while eax < MAX_LEN
        .if OriginMapText[eax * 4] == 5
            ;如果Origin是5的位置 Current都是4就行了
            .if CurrentMapText[eax * 4] == 4
            jmp L1
            .else ;不等于4,说明没成功
                jmp NotWin
            .endif
        .endif
L1:     inc eax
```

```
    .endw
    mov isWin, 1 ;该局获胜
    mov ebx, CurrBestLevel
    .if currentLevel == ebx
        inc CurrBestLevel
    .endif
    inc currentLevel ;关卡数+1
    ;invoke MessageBox, NULL, addr szSuccess, addr szTitle, MB_OK

    ret
NotWin:
    mov isWin, 0
    ret

    JudgeWin endp
```

4-4

第 5 章 功能验证与测试

在本游戏编写完成后，对游戏的具体功能是否成功得以实现进行了测试工作以及功能验证。具体测试内容如下：

1.游戏是否能正常启动，窗口显示是否正常。

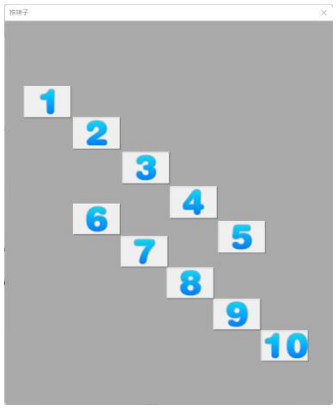
测试方式：打开程序，观看是否有窗口弹出。

测试结果：游戏可以正常启动并显示开始界面，测试通过。

2.游戏能否正常开始。

测试方式：单机开始按钮，观察游戏是否进行。

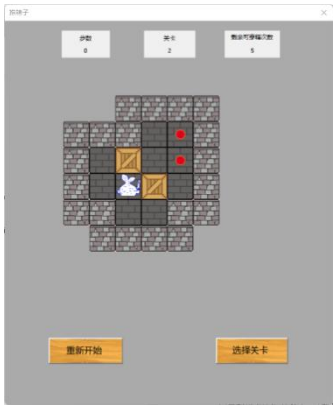
测试结果：游戏可以正常开始，并且显示选关界面，测试通过。



3.游戏关卡内容能否正常显示。

测试方式：选择任一关卡打开，查看是否弹出关卡界面。

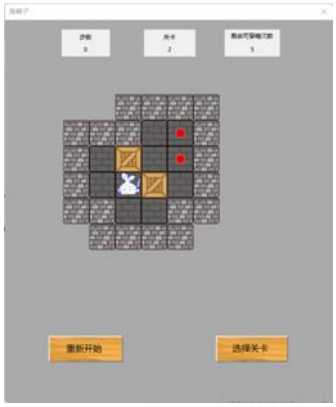
测试结果：关卡内容正常显示，测试通过。



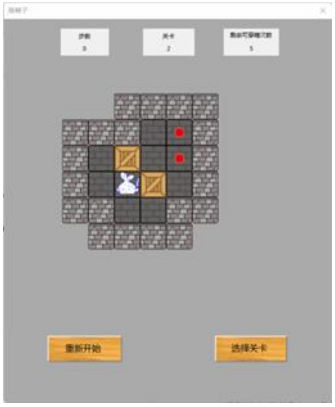
4.人物以及箱子，墙壁等 ui 是否能够正常显示。

测试方式：选择任一关卡打开，在关卡中检查人物以及墙壁等的 ui。

测试结果：ui 可以正常显示出来，测试通过。

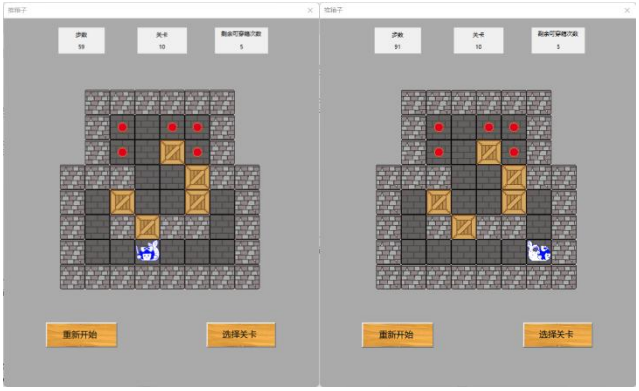


5.上方关卡记录数字是否正确。
测试方式：选择任一关卡打开，观察上方关卡记录是否和选择的关卡一致。
测试结果：上方关卡记录和选择的关卡一致，测试通过。

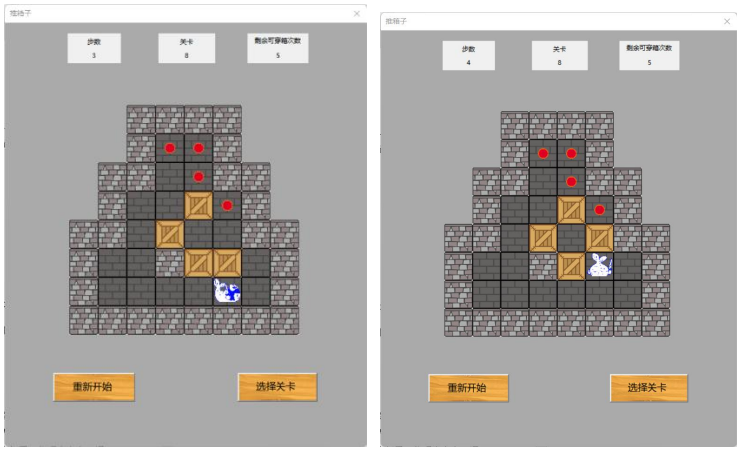


6.人物移动功能是否正常。
测试方式：通过上下左右键移动人物，观察人物在界面上是否移动并且更改朝向。
测试结果：人物可以根据键盘输入的指令进行移动，并且更改人物图片，测试通过。

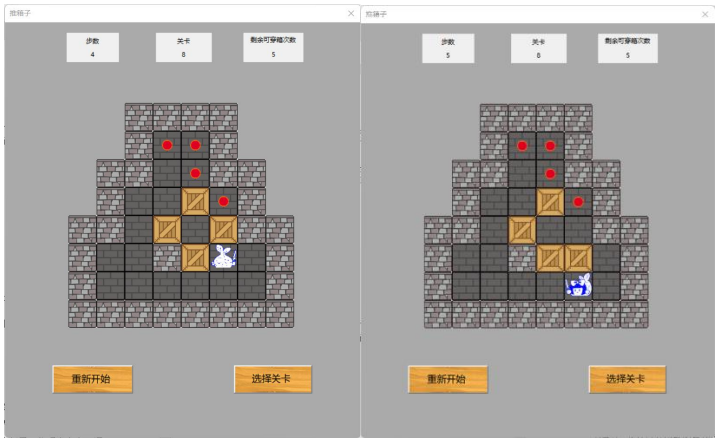
7.上方步数记录功能能否正常记录人物移动的步数。
测试方式：输入指令让人物移动，观察上方步数记录是否有变化。
测试结果：上方步数记录会随着人物的移动发生变化，可以正常记录人物的移动，测试通过。



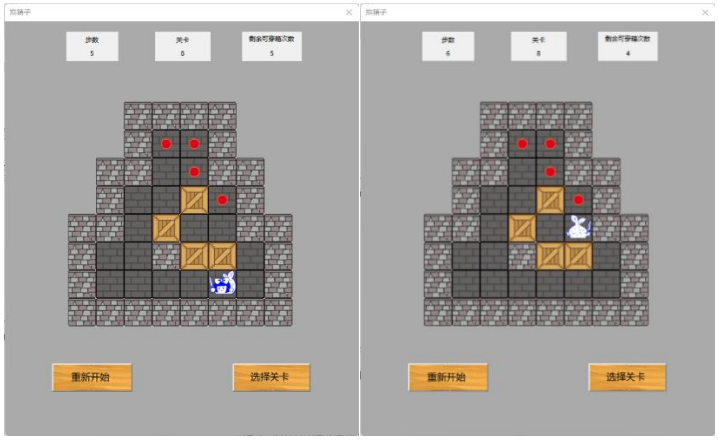
8. 人物在接触到箱子时，能否正常的将箱子推走。
测试方式：让人物紧贴箱子进行移动，并且确保箱子后方存在空位，观察箱子是否跟随人物一起移动。
测试结果：当人物移动时，箱子可以跟随人物一起移动，测试通过。



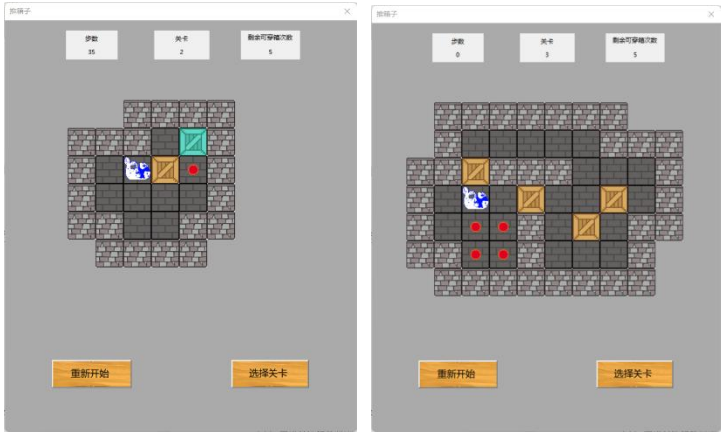
9. 人物在使用“拉”箱子功能时，能否正常的将箱子拉开。
测试方式：让人物紧贴箱子进行 CTRL+移动键进行移动，并且确保箱子后方存在空位，观察箱子是否跟随人物一起移动。
测试结果：当人物使用 CTRL+移动键进行移动时，箱子可以跟随人物一起移动，测试通过。



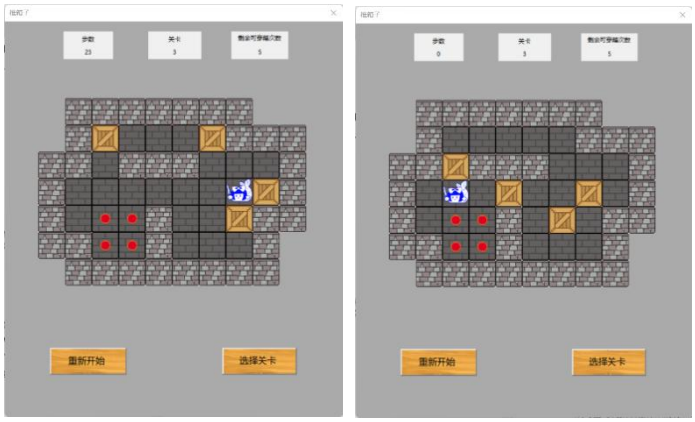
10. 人物在使用“穿”箱子功能时，能否正常的穿过箱子进行移动。
测试方式：让人物紧贴箱子进行 SHIFT+移动键进行移动，并且确保箱子后方存在空位，观察人物是否成功穿过箱子，同时右上方记录是否正常扣除次数。
测试结果：当人物使用 SHIFT+移动键进行移动时，人物可以穿过箱子进行移动，右上方穿箱子次数正常扣除，测试通过。



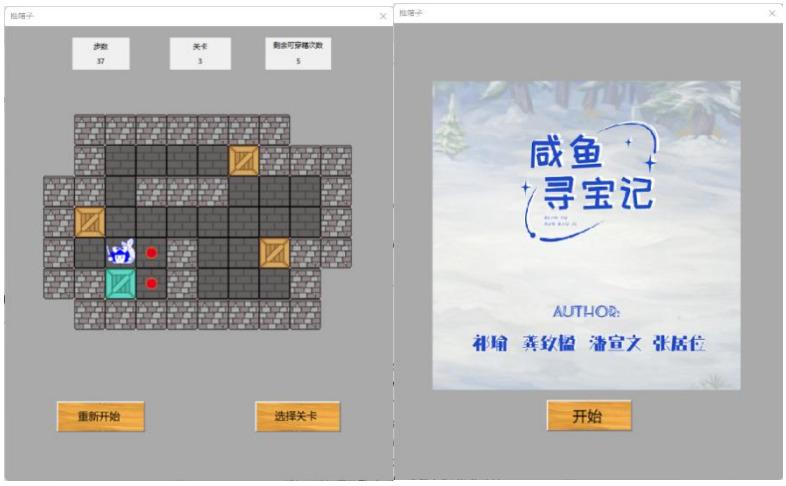
11. 游戏是否能够正常通关。
测试方式：选择任一关卡正常进行游戏，将全部箱子推至判定点后，观察游戏是否成功进入到下一关。
测试结果：当将全部箱子推至判定点后，游戏顺利进入到下一关，测试通过。



12.下方重新开始按钮是否正常工作。
测试方式：选择任一关卡正常进行游戏，进行移动打乱原顺序后，点击重新开始按钮。
测试结果：点击重新开始按钮后，关卡恢复正常排列，正常重新开始，上方步数记录也成功归零，测试通过。



13.下方选择关卡按钮是否正常工作。
测试方式：选择任一关卡正常进行游戏，进行移动打乱原顺序后，点击选择关卡按钮。
测试结果：点击选择关卡按钮后，重新返回关卡选择界面，测试通过。



人员分工

祁瑜：组长，窗口基本函数、人物移动功能、拉箱子功能编写
潘宣文：演示 ppt 制作、UI 设计、关卡设计、地图生成函数编写
龚致楹：按钮事件响应、界面样式调整、控件布局调整
张居位：游戏结束判断函数、穿越箱子功能编写、程序功能测试

结　论

本次汇编语言游戏程序是一次对汇编理论知识的实践。我们通过汇编语言编写了经典游戏推箱子。相比传统推箱子游戏，我们增加了丰富的游戏玩法，添加了拉

箱子和穿越箱子功能，并设计了对应必须使用该功能的地图。游戏更加新颖，可玩性增加。对于技能的使用有次数限制，适当增大游戏难度，增加游戏挑战性。优化游戏界面，使用更有趣的 UI，增加游戏兴趣。在实践过程中，我们探索学习了 windows 窗口程序编程规范，明白了消息循环，鼠标和键盘事件的处理过程；学习了 masm32 汇编伪指令，简化了游戏开发；使用位图在窗口中显示图片，美化图形界面；熟悉了汇编运算、函数调用的过程。同时团队分工锻炼了大家的沟通合作能力，在交流过程中大家取长补短共同进步。

致　谢

经过一学期努力，汇编语言课程终于完成。首先向张华平教授表示深深感谢，在课程中悉心指导我们学习知识，提供给我们团队合作完成实践的机会，提高知识水平、锻炼合作能力。也感谢课代表的付出、同班同学们的陪伴以及交流，创造了良好的学习环境，拓展了思维。