北京理工大學

汇编语言与接口技术

小组上机作业实验报告

Report

学　　院：计算机学院

专　　业：计算机科学与技术

学生姓名：刘嘉诚、张成吉吉、常诗冉、崔晨曦

学　号：1120180345、1120181129、1120180480、1120181319

指导教师：李元章

2021 年 7 月 6 日

目　录

第 1 章 项目概况

本项目采用 32 位汇编语言进行了基于 MASM32 的游戏开发，首先实现了 Flappy Bird 小游戏的基本功能，然后不断地迭代，进行了游戏难度升级，成功加入了双人对战、大小招、奖惩机制等复杂模式，实现了最终的双人对战呆鸟游戏。

第 2 章 项目设计

2. 1 设计灵感与项目背景

本项目的设计灵感来源于曾经红极一时的手机游戏 Flappy Bird，它由越南独立游戏开发者 Dong Nguyen 开发。这是一款兼具简单性和困难性的游戏，简单在游戏玩家只需要一只手指操控，困难在玩家必须控制一只胖乎乎的小鸟跨越不同长度的水管，需要恒心和耐力。

本项目追求一定的复杂性，因此在用 32 位汇编语言实现电脑版的基本功能后，进一步迭代进行了功能和模式设计，加入了许多新颖亮眼的巧思，譬如双人呆鸟对战，Debuff 机制、奖惩机制等，让游戏的趣味性和项目的价值大大提升。

2. 2 基本操作

在本游戏项目中，小鸟以固定速度相对于柱子移动，玩家则通过键盘按键控制小鸟上下移动，有以下几个主要操作：

2. 2. 1 控制升降：空格 Space、X

玩家通过按空格键或 X 键分别控制第一只小鸟和第二只小鸟，默认状态下小鸟会以一定的重力速度下降，当按下 Space 和 X 键后，小鸟会获得一定大小的向上加速度并上移。

2. 2. 2 控制小招：按键 M、Z

当玩家连续飞过 10 个柱子后，获得小招奖励，可以通过键盘控制按动 M 键或者 Z 键在关键时刻释放小招，给对手加上 Debuff。

2. 2. 3 控制大招：按键 L、Q

当晚间连续飞过 30 根柱子后可以续满一个大招，通过按动键盘上的 L 键或者 Q 键进行释放。

2. 3 对战模式

若小鸟触底或撞到管子上，判定为死亡，失去生命。

双人呆鸟对战模式下，每只小鸟拥有 5 条生命值，首先失去五条生命的小鸟为输家，另一方则成为赢家。

2. 4 对战机制

对战机制可以总结为三条理念

- 操作失误自己接受惩罚
- 没有失误对方接受乘法
- 接受惩罚后仍有翻盘的机会

具体来说，当某个玩家的小鸟连续飞过了 10 根柱子后可以获得小招，在一定时刻控制释放给对手加 Debuff。当小鸟撞死复活后也要自身接受 Debuff。这里所谓的 Debuff 是一种惩罚机制，被施加的小鸟朝柱子横向飞翔的速度会加快，纵向向下加速度会加大升起更慢，小鸟会变绿作为标识。Debuff 的效果会随时间减弱至无。如果某个玩

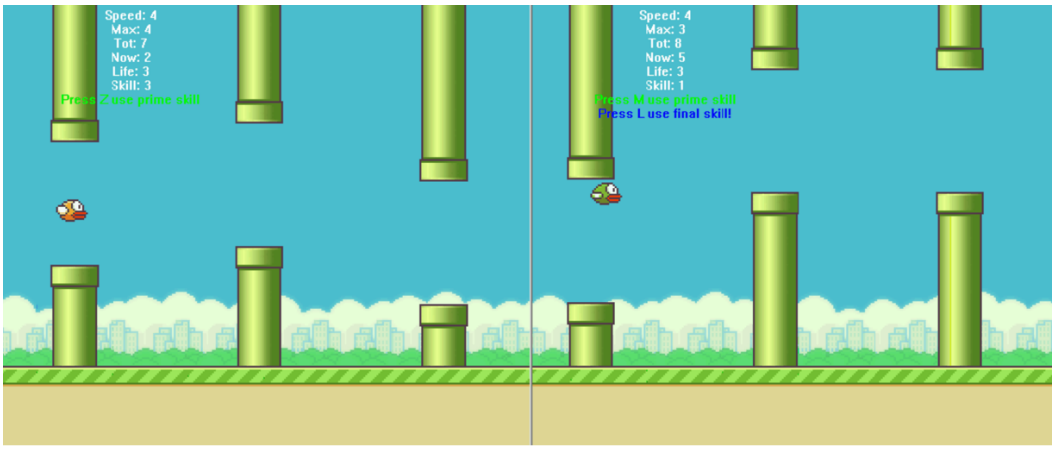

图 2-1 Debuff 机制示意图

家的小鸟连续飞过 30 根柱子，那么可以获得一次大招并自行选择在某一时刻释放，释放大招的小鸟自身会拥有一段“无敌期”，随后上下柱子间宽度会突然变到很大，然后随时间减少至正常状态，此过程持续 10 秒。

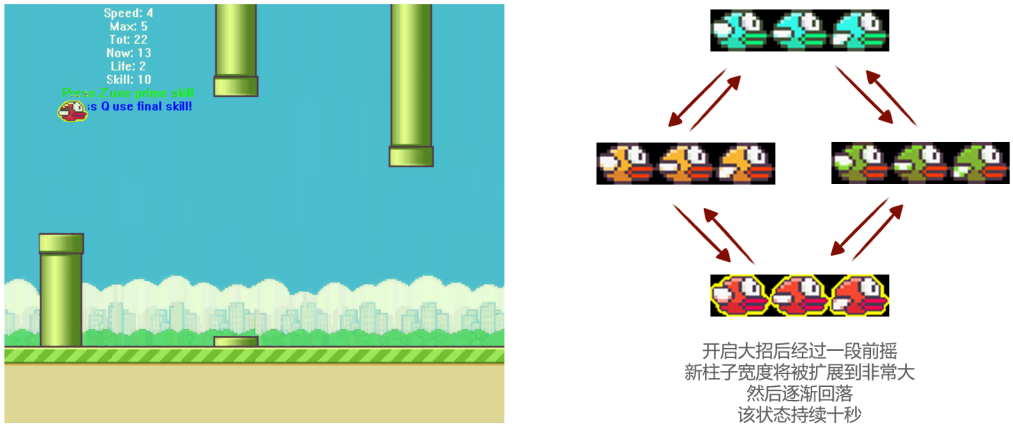
Speed: 4
Max: 5
Tot: 22
Now: 13
Life: 2
Skill: 10
Press Z use prime skill!
s Q use final skill!
开启大招后经过一段前摇
新柱子宽度将被扩展到非常大
然后逐渐回落
该状态持续十秒

图 2-2 大招机制示意图

第 3 章 实验环境

- 汇编语言编写环境：Microsoft Visual Studio Community 2019
- 汇编编译器：MASM32 SDK

第 4 章 项目实现

4.1 程序主体框架

项目主要包含 AsmFlappyBird.Inc 和 AsmFlappyBird.Asm 两个文件。

4.1.1 AsmFlappyBird.Inc

AsmFlappyBird.Inc 对所要用到的导入库、库函数、Windows API 函数等进行了预先声明，用伪指令进行了必要的符号定义，声明了数据结构和程序中将要使用到的变量。

例如，记录柱子位置和宽度信息的数据结构如下所示：

Listing 4.1: 记录柱子信息的数据结构

```
1 OBJECT  struct
2   x    dd ?
3   y    dd ?
4   wide   dd ?
5 OBJECT ends
```

其中 x 是柱子的横坐标，y 是柱子的纵坐标，wide 是上下两根柱子之间的开口宽度。

其他的变量将在后续的程序说明中给出具体的介绍。

4.1.2 AsmFlappyBird.Asm

AsmFlappyBird.Asm 是项目的主体部分，主要包含以下几个功能函数：

- WndProc：加载窗口界面元素，初始化，接收处理玩家键盘操作
- TimerProc：实现重力作用的下降、碰撞检验、记分、创建新柱子和物体移动
- BoardProc(2)、PaintBoard(2)：绘制界面
- DeBuff2：实现在指定的小鸟上施加 Debuff 惩罚，小鸟变重，且横向移动加快。
- ResetBuff：控制某次 Debuff 效果逐渐减弱直至无
- EndGame：初始化游戏，或在游戏结束后恢复到初始状态
- SetupFly、InitParam：初始化鸟和柱子的位置、初始化游戏参数
- SetupDie2：小鸟死亡后，记录最后一次的得分、最大得分，并将分数归零，恢复小鸟和柱子的位置

- Random： 随机数生成

4.2 窗口绘制

首先，按照标准写法初始化窗口的各个属性:

Listing 4.2: 初始化窗口属性

```
; 窗口主程序
WinMain proc hInst:HINSTANCE, hPrevInst:HINSTANCE, CmdLine:LPSTR, CmdShow:DWORD
  LOCAL wc:WNDCLASSEX
  LOCAL msg:MSG
    ;窗口属性

  mov    wc.cbSize, sizeof WNDCLASSEX
  mov    wc.style, CS_HREDRAW or CS_VREDRAW
  mov    wc.lpfnWndProc, offset WndProc
  mov    wc.cbClsExtra, NULL
  mov    wc.cbWndExtra, DLGWINDOWEXTRA
  push   hInst
  pop    wc.hInstance
  mov    wc.hbrBackground, COLOR_BTNFACE+1
  mov    wc.lpszMenuName, IDM_MENU
  mov    wc.lpszClassName, offset ClassName
  invoke LoadIcon, NULL, IDI_APPLICATION
  mov    wc.hIcon, eax
  mov    wc.hIconSm, eax
  invoke LoadCursor, NULL, IDC_ARROW
  mov    wc.hCursor, eax

  invoke RegisterClassEx, addr wc
  invoke CreateDialogParam, hInstance, IDD_DIALOG, NULL, addr WndProc, NULL
  invoke ShowWindow, hWnd, SW_SHOWNORMAL
  invoke UpdateWindow, hWnd

    ;mov isFirst, 1
  .while TRUE
    invoke GetMessage, addr msg, NULL, 0, 0
    .BREAK .if !eax
    invoke TranslateMessage, addr msg
    invoke DispatchMessage, addr msg
```

```
34    .endw
35    mov     eax , msg.wParam
36    ret
37
38  WinMain  endp
```

4. 2. 1 基本组件绘制

采用逐帧打印的方式显示窗口界面，每 25 毫秒打印一次。主要通过 WndProc 函数初始化、TimerProc 函数控制刷新。

通过 Win32 子窗口实现双人对战模式下的游戏界面，如下图所示：首先，在

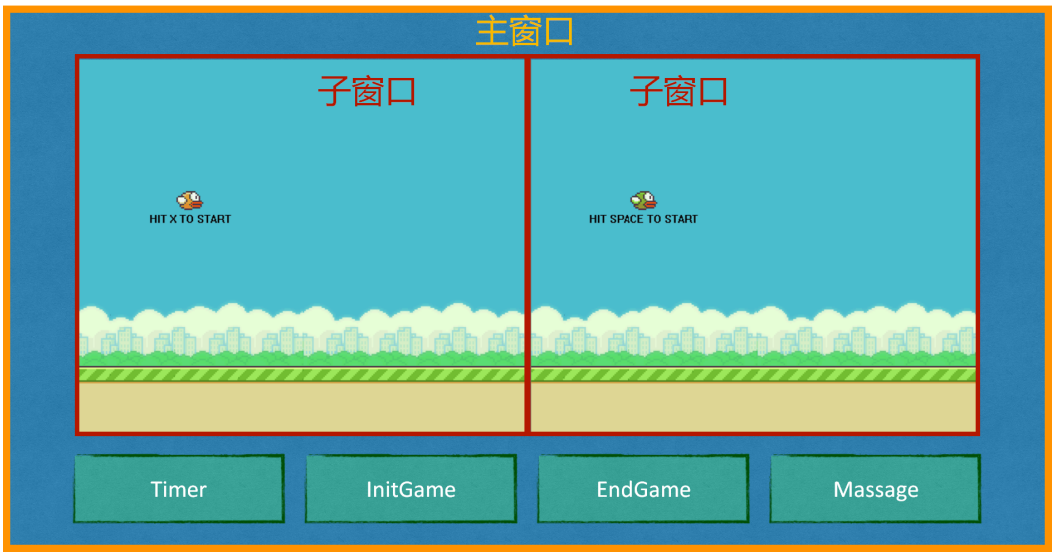

图 4-1 双人对战游戏界面

WinProc 函数中加载进背景、管子、Play1 小鸟，Play2 小鸟、debuff 状态下的小鸟、大招状态下的小鸟、地面这些窗口界面组成元素的照片。例如：

Listing 4.3: 加载窗口界面组成元素

```
1  ; Load bird_de image
2  invoke ImageList_Create ,34 ,24 ,ILC_COLOR16 or ILC_MASK,3 ,3
3  mov    birdIml3 , eax
4  invoke LoadBitmap , hInstance , IDB_BIRD3
5  mov    hBmp, eax
6  invoke ImageList_AddMasked , birdIml3 , hBmp,0
7  invoke DeleteObject , hBmp
```

然后，在 BoardProc 和 BoardProc2 函数中分别调用 PaintBoard 和 PaintBoard2 函数中绘制两个小鸟的界面，具体绘制方法会在下面的模块里展开叙述，在此不赘述。

Listing 4.4: BoardProc 函数

```
BoardProc proc hWin:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM
  LOCAL ps:PAINTSTRUCT

  .if uMsg==WM_PAINT
    invoke BeginPaint,hWin,addr ps
    invoke PaintBoard,hWin,ps.hdc
    invoke EndPaint,hWin,addr ps
    xor   eax,eax
    ret
  .endif
  invoke CallWindowProc,OldBoardProc,hWin,uMsg,wParam,lParam
  ret

BoardProc endp
```

接着，调用 EndGame 函数进行初始化，将鸟的位置设置在初始化界面中间，鸟的生命值初始化为 5，并调用 InitParam 函数、SetupFly 函数，前者初始化游戏中鸟的位置和柱子的位置，后者初始化游戏中的各种参数，比如速度、生命数、加速度等。

初始化界面效果如下图所示:

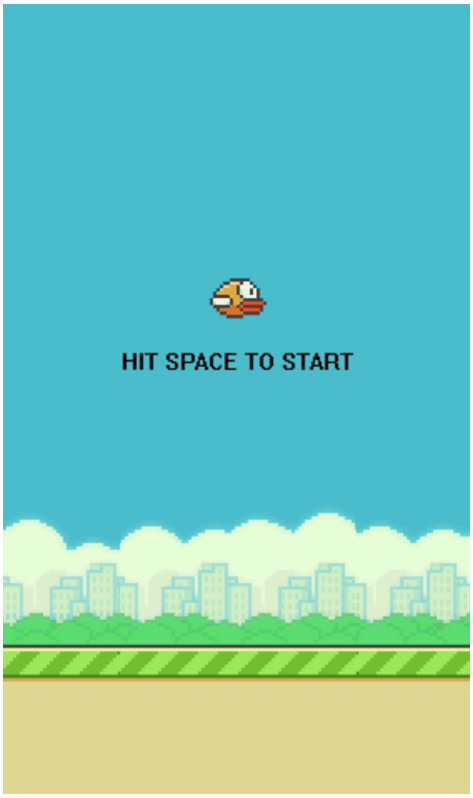

图 4-2 初始界面

4.2.2 柱子移动与切换

对于每只小鸟维护两组柱子，每组包含上下两个柱子，每个柱子包含 x 和 y 坐标，开口位置随机选择，总共维护 4 对柱子，两组柱子轮换。

初始化柱子在 WndProc 函数中实现，部分代码如下：

Listing 4.5: 初始化柱子

```
; Draw tube here
push ebx

mov ebx, offset upTube2
.if [ebx].OBJECT.x < 288 || [ebx].OBJECT.x <= -1
invoke ImageList_Draw, tubeIml, 0, mDC, [ebx].OBJECT.x, [ebx].OBJECT.y,
    ILD_TRANSPARENT
.endif
add ebx, sizeof OBJECT
.if [ebx].OBJECT.x < 288 || [ebx].OBJECT.x <= -1
invoke ImageList_Draw, tubeIml, 0, mDC, [ebx].OBJECT.x, [ebx].OBJECT.y,
    ILD_TRANSPARENT
```

```
11  .endif
12  add ebx, sizeof OBJECT
13  .if [ebx].OBJECT.x < 288 || [ebx].OBJECT.x <= -1
14  invoke ImageList_Draw, tubeIml, 0, mDC, [ebx].OBJECT.x, [ebx].OBJECT.y,
        ILD_TRANSPARENT
15  .endif
16  add ebx, sizeof OBJECT
17  .if [ebx].OBJECT.x < 288 || [ebx].OBJECT.x <= -1
18  invoke ImageList_Draw, tubeIml, 0, mDC, [ebx].OBJECT.x, [ebx].OBJECT.y,
        ILD_TRANSPARENT
19  .endif
20
21  mov ebx, offset downTube2
22  .if [ebx].OBJECT.x < 288 || [ebx].OBJECT.x <= -1
23  invoke ImageList_Draw, tubeIml, 1, mDC, [ebx].OBJECT.x, [ebx].OBJECT.y,
        ILD_TRANSPARENT
24  .endif
25  add ebx, sizeof OBJECT
26  .if [ebx].OBJECT.x < 288 || [ebx].OBJECT.x <= -1
27  invoke ImageList_Draw, tubeIml, 1, mDC, [ebx].OBJECT.x, [ebx].OBJECT.y,
        ILD_TRANSPARENT
28  .endif
29  add ebx, sizeof OBJECT
30  .if [ebx].OBJECT.x < 288 || [ebx].OBJECT.x <= -1
31  invoke ImageList_Draw, tubeIml, 1, mDC, [ebx].OBJECT.x, [ebx].OBJECT.y,
        ILD_TRANSPARENT
32  .endif
33  add ebx, sizeof OBJECT
34  .if [ebx].OBJECT.x < 288 || [ebx].OBJECT.x <= -1
35  invoke ImageList_Draw, tubeIml, 1, mDC, [ebx].OBJECT.x, [ebx].OBJECT.y,
        ILD_TRANSPARENT
36  .endif
37
38  pop ebx
39  ; end draw tube
```

通过 TimerProc 函数控制，柱子以设定的速度 speed 移动，改变的是柱子的 x 坐标，上下两根柱子要同时移动。移动柱子部分的代码如下：

Listing 4.6: 控制柱子移动

```
mov ecx, 4
        mov ebx, offset upTube
        mov edx, offset downTube

        ;移动柱子位置
        .while ecx
            mov eax, [ebx].OBJECT.x
            sub eax, speed
            mov [ebx].OBJECT.x, eax
            mov [edx].OBJECT.x, eax
            add ebx, sizeof OBJECT
            add edx, sizeof OBJECT
            dec ecx
        .endw
        pop edx
        pop ebx
```

当一个柱子移出窗口外，在 TimerProc 函数中创建新柱子。即当最左边的柱子移出窗口外，按顺序将 2 号柱子转移到 1 号柱子，将 3 号柱子移到 2 号柱子，将 4 号柱子移到 3 号柱子；上下柱子做同样的操作。然后再创建一对新的柱子作为新的 4 号柱子。代码实现如下：

Listing 4.7: 创建新柱子

```
.if [ebx].OBJECT.x < -54 && [ebx].OBJECT.x > 10000
            mov eax, ebx
            add eax, sizeof OBJECT
            mov ecx, [eax].OBJECT.x
            mov [ebx].OBJECT.x, ecx     ;2号柱子转移到1号柱子
            mov ecx, [eax].OBJECT.y
            mov [ebx].OBJECT.y, ecx
            mov ecx, [eax].OBJECT.wide
            mov [ebx].OBJECT.wide, ecx

            mov ebx, eax
            add eax, sizeof OBJECT
            mov ecx, [eax].OBJECT.x
            mov [ebx].OBJECT.x, ecx     ;3号柱子转移到2号柱子
            mov ecx, [eax].OBJECT.y
            mov [ebx].OBJECT.y, ecx
            mov ecx, [eax].OBJECT.wide
```

```
18              mov [ebx].OBJECT.wide, ecx
19
20              mov ebx, eax
21              add eax, sizeof OBJECT
22              mov ecx, [eax].OBJECT.x
23              mov [ebx].OBJECT.x, ecx      ;4号柱子转移到3号柱子
24              mov ecx, [eax].OBJECT.y
25              mov [ebx].OBJECT.y, ecx
26              mov ecx, [eax].OBJECT.wide
27              mov [ebx].OBJECT.wide, ecx
28
29              mov eax, edx
30              add eax, sizeof OBJECT
31              mov ecx, [eax].OBJECT.x
32              mov [edx].OBJECT.x, ecx
33              mov ecx, [eax].OBJECT.y
34              mov [edx].OBJECT.y, ecx
35              mov ecx, [eax].OBJECT.wide
36              mov [edx].OBJECT.wide, ecx
37
38              mov edx, eax
39              add eax, sizeof OBJECT
40              mov ecx, [eax].OBJECT.x
41              mov [edx].OBJECT.x, ecx
42              mov ecx, [eax].OBJECT.y
43              mov [edx].OBJECT.y, ecx
44              mov ecx, [eax].OBJECT.wide
45              mov [edx].OBJECT.wide, ecx
46
47              mov edx, eax
48              add eax, sizeof OBJECT
49              mov ecx, [eax].OBJECT.x
50              mov [edx].OBJECT.x, ecx
51              mov ecx, [eax].OBJECT.y
52              mov [edx].OBJECT.y, ecx
53              mov ecx, [eax].OBJECT.wide
54              mov [edx].OBJECT.wide, ecx
55
56
57              mov eax, [edx].OBJECT.x
```

```
58              add eax, 200
59              add ebx, sizeof OBJECT
60              add edx, sizeof OBJECT
61              mov [ebx].OBJECT.x, eax
62              mov [edx].OBJECT.x, eax
63              mov ecx, inFinSkill
64              mov [ebx].OBJECT.wide, ecx
65              mov [edx].OBJECT.wide, ecx
```

4.2.3 鸟的振翅

为了提升游戏的趣味性和逼真性，我们在窗口界面中模拟了小鸟的振翅动作。预先处理了了小鸟振翅不同状态的照片，然后按 0,1,2,2,0,1,2 的方式对图片进行切换，从而实现小鸟振翅的效果。

控制部分的代码如下所示：

Listing 4.8: 控制小鸟振翅

```
1 ; Draw bird
2 .if cflap < 1 && cflapDir == 3
3 mov cflapDir, 0
4 .elseif cflap > 1 && cflapDir == 3
5 mov cflapDir, 1
6 .elseif cflap < 1 || cflap > 1
7 mov cflapDir, 3
8 .endif
```

小鸟的振翅原理图及效果如下所示：

4.2.4 底部边框滑动

至于底部边框滑动的控制，首先将图片左边界和窗口左边界对齐，从右往左移动，每当图片右边界与窗口右边界对齐，则将底部边框图片整个后移至图片左边界和窗口左边界对齐，重复上述过程，让底部边框一直可以保持滑动效果。

具体控制部分代码如下所示：

Listing 4.9: 控制底部边框滑动

```
1 ; Draw bottom
2 mov eax, speed2
3 sub bottomX2, eax
4 .if bottomX2 < -47
```

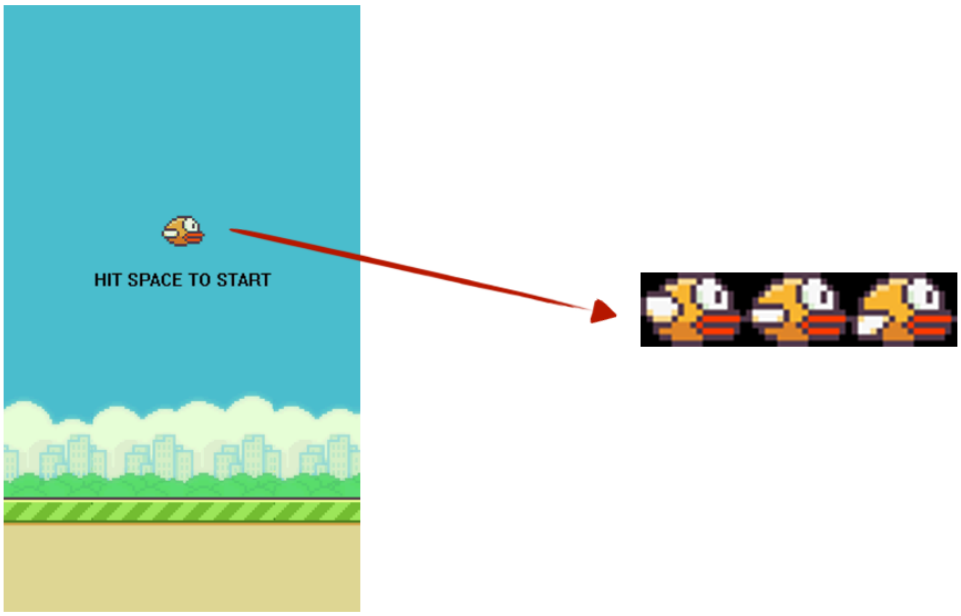

图 4-3 小鸟振翅原理图

```
5  mov bottomX2, 0
6  .endif
7  invoke ImageList_Draw, barIml, 0, mDC, bottomX2, bottomY2, ILD_TRANSPARENT
8
9  invoke GetClientRect,hWin,addr rect
10 invoke StretchBlt,hDC,0,0,rect.right,rect.bottom,mDC,0,0,288+288,500,SRCCOPY
11 pop   eax
12 invoke SelectObject,mDC,eax
13 invoke DeleteObject,eax
14 invoke DeleteDC,mDC
15 ret
```

底部边框滑动的原理和效果如下图所示：

4.3 控制与判断

4.3.1 鸟的操控

玩家通过键盘按键控制鸟向上飞、释放小招、释放大招。

这一部分操作在 WinProc 函数中进行处理，代码如下：

Listing 4.10: 操控小鸟的移动

```
1
```

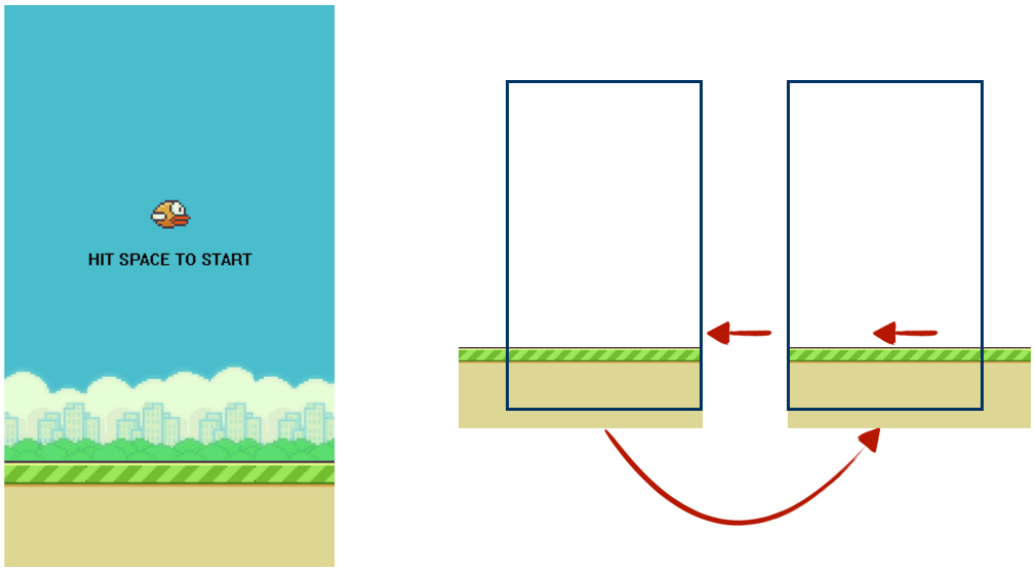

图 4-4 底部边框滑动原理图

```
    LOCAL  hBmp:DWORD
mov     eax,uMsg

    ;WM_INITDIALOG消息是对话框才能收到的消息，表明对话框及其所有子控件都创建完毕了
.if  eax==WM_INITDIALOG
    ......
.elseif  eax==WM_KEYDOWN
    mov  eax,wParam
    .if  eax==VK_SPACE
        .if  gStatus == 0
            mov  gStatus, 1
            invoke  SetupFly
        .else
            mov  eax, jumpAc2
            mov  birdAc2, eax
        .endif

            .if  isFirst == 1
                mov  isFirst,0
            .endif

        ;new
```

```
24          .elseif eax==VK_X
25      .if gStatus == 0
26          mov gStatus, 1
27          invoke SetupFly
28      .else
29          mov eax, jumpAc
30          mov birdAc, eax
31      .endif
32
33              .if isFirst == 1
34                  mov isFirst,0
35              .endif
36          ;new
37          .elseif eax==VK_M
38              .if gStatus == 1
39            .if Disturb2 > 0
40                      dec Disturb2
41                      invoke DeBuff2, 1
42                  .endif
43      .endif
44
45          .elseif eax==VK_Z
46              .if gStatus == 1
47            .if Disturb > 0
48                      dec Disturb
49                      invoke DeBuff2, 2
50                  .endif
51      .endif
52
53          .elseif eax==VK_L
54              .if gStatus == 1
55            .if FinDisturb2 > 0
56                      mov inFinSkill2, 400
57                      ; TODO
58                      ;mov inFinSkill, -30
59                      mov FinDisturb2, 0
60                  .endif
61      .endif
62
63          .elseif eax==VK_Q
```

```
64              .if gStatus == 1
65            .if FinDisturb > 0
66                      mov inFinSkill, 400
67                      ; TODO
68                      ;mov inFinSkill2, -30
69                      mov FinDisturb, 0
70                  .endif
71        .endif
72          .endif
73
74    .elseif eax==WM_COMMAND
75      mov     eax,wParam
76      and     eax,0FFFFh
77      .if eax==IDM_FILE_EXIT
78        invoke SendMessage,hWin,WM_CLOSE,0,0
79      .elseif eax==IDM_SAVE
80          invoke SaveFile
81      .elseif eax==IDM_OPEN
82          invoke ReadModelFile
83      .endif
84
85    .elseif eax==WM_CLOSE
86      invoke DestroyWindow,hWin
87
88    .elseif uMsg==WM_DESTROY
89      invoke PostQuitMessage,NULL
90
91    .else
92      invoke DefWindowProc,hWin,uMsg,wParam,lParam
93      ret
94    .endif
```

即，对于 Play1 小鸟、Play2 小鸟，分别通过空格和 X 控制向上飞、按动 M 和 Z 向对手释放 Debuff 技能，按动 L 和 Q 释放大招。

小鸟相对于柱子以一个固定的速度 Speed 横向移动，纵向受重力 GravityAc 的影响自然下落，每当玩家按下空格 Space 或 X 键，小鸟会当即获得一个上升力 JumpAc，重力和上升力叠加成为小鸟的速度 BirdAc。但最大下落速度有限制。

示意图如下：

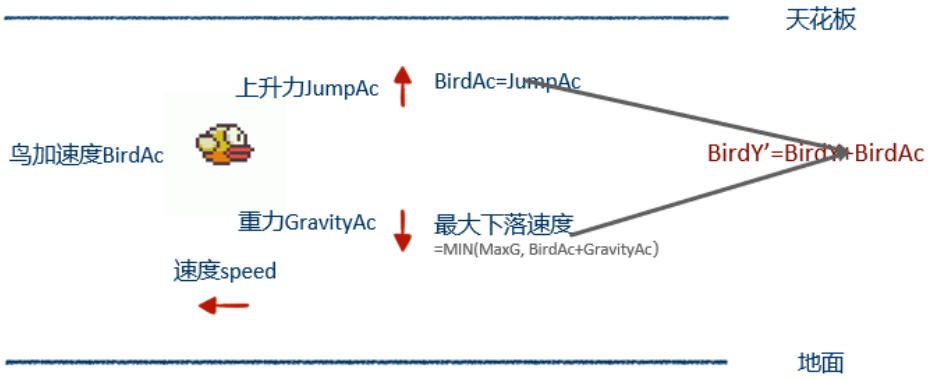

图 4-5 鸟的控制示意图

4.3.2 死亡判断

呆鸟有两种死亡的情况：触地、撞到管子，在 TimeProc 函数里实现碰撞检测。如果小鸟的纵坐标超过地面的纵坐标，判定为触地。

Listing 4.11: 判定触地

```
1 .if eax > birdBottom ; touch the ground, failed
2   invoke SetupDie2,1
```

碰撞检测示意图如下，当小鸟的横坐标在第一个柱子宽度范围内，进一步判定高度的关系，如果小鸟的纵坐标高于上柱子的下界、或低于下柱子的上届，则代表小鸟碰上了柱子，调用 SetupDie 函数进行后续处理。

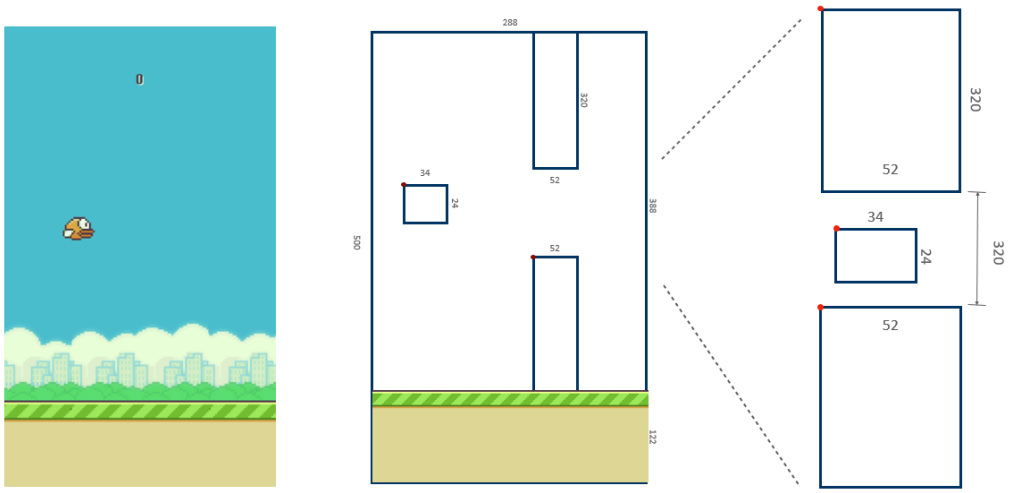

图 4-6 碰撞检测示意图

判定部分代码如下：

Listing 4.12: 碰撞检测部分代码

```
1
2       mov ebx, offset upTube
3       mov edx, offset downTube
4       .if eax > 16 && eax < 97
5           mov eax, [edx].OBJECT.y
6           sub eax, birdY
7           .if eax < 20 || eax > 10000
8               invoke SetupDie2,1
9           .else
10              mov eax, birdY
11              sub eax, [ebx].OBJECT.y
12              .if eax < 316 || eax > 10000
13                  invoke SetupDie2,1
14              .endif
15          .endif
16      .endif
```

如果小鸟的横坐标超过了第一个柱子的范围，且没有进行过加分，则当前分数加一。

在 TimerProc 函数里进行记分：

Listing 4.13: 记分

```
1       .if eax < birdX && addedFlag == 0
2           invoke ResetBuff,1
3           inc cPoint
4           mov eax, cPoint
5           mov edx, 0
6           mov ebx, DisturbMod
7           div ebx
8           .if edx == 0
9               inc Disturb
10          .endif
11          mov eax, cPoint
12          mov edx, 0
13          mov ebx, FinDisturbMod
14          div ebx
15          .if edx == 0
```

```
16                  mov FinDisturb, 1
17              .endif
18              inc totPoint;new
19              mov addedFlag, 1
20          .endif
```

4.4 技能与 Buff

4.4.1 DeBuff 的施加与恢复

在某玩家的小鸟连续飞过 10 根柱子后，即可获得一次给对手施加 Debuff 的机会，获得后玩家可以通过按对应的 M（玩家 1）或 Z（玩家 2）键释放。

Listing 4.14: 按键释放 Debuff

```
1     .elseif eax==VK_M
2         .if gStatus == 1
3         .if Disturb2 > 0
4                 dec Disturb2
5                 invoke DeBuff2, 1
6             .endif
7     .endif
8
9     .elseif eax==VK_Z
10        .if gStatus == 1
11        .if Disturb > 0
12                dec Disturb
13                invoke DeBuff2, 2
14            .endif
15    .endif
```

小鸟死亡复活后此机制也会被触发，需要自身承受一次 Debuff 惩罚。

释放或触发后调用 Debuff2 函数产生惩罚效果，被施加的小鸟朝柱子横向飞翔的速度会加快，纵向向下加速度会加大，对应着向上跳跃的速度更慢，小鸟会变绿作为标识。

Listing 4.15: 施加 Debuff

```
1
2   DeBuff2 proc target:dword
3     ;Play 1
4     .if target==1
```

```
        mov indebuf,1   ;debuf标记置位
        mov speed, 5    ;加速
        mov gravityAc, 2 ;增加重力
        mov jumpAc, -12  ;减少跳跃力度
    .endif
  ;Play 2
    .if target==2
        mov indebuf2,1
        mov speed2, 5
        mov gravityAc2, 2 ;增加重力
        mov jumpAc2, -12  ;减少跳跃力度
    .endif
    ret
DeBuff2 endp
```

Debuff 的效果会随时间减弱至无，这一过程由 ResetBuff 函数控制。通过传入的 target 参数辨别玩家 1 和玩家 2。

Listing 4.16: Debuff 效果逐渐减弱至无

```

ResetBuff proc target:dword
  ;Play 1
    .if target==1
        .if indebuf==1
            .if speed>4
                dec speed
            .elseif
                mov gravityAc, 2
                mov jumpAc, -14
                mov indebuf,0
            .endif
        .endif
    ;Play 2
    .elseif target==2
        .if indebuf2==1
            .if speed2>4
                dec speed2
            .elseif
                mov gravityAc2, 2
                mov jumpAc2, -14
```

```
22                  mov indebuf2 ,0
23              .endif
24          .endif
25      .endif
26      ret
27  ResetBuff endp
```

4.4.2 终极技能

如果某个玩家的小鸟连续飞过 30 根柱子，那么可以获得一次大招并自行选择在某一时刻释放，释放大招的小鸟自身会拥有一段“无敌期”，随后上下柱子间宽度会突然变到很大，然后随时间减少至正常状态，此过程持续 10 秒。

为了实现控制，小鸟获得大招并按下 L（玩家 1）或 Q（玩家 2）释放后，首先在 WinProc 函数将 inFinSkill2 的变量设置成 400，将 FinDisturb2 的值设为 0，以供后续操作。

Listing 4.17: 释放大招

```
1   ;Play 1
2   .elseif eax==VK_L
3       .if gStatus == 1
4       .if FinDisturb2 > 0
5               mov inFinSkill2, 400
6               ; TODO
7               ;mov inFinSkill, -30
8               mov FinDisturb2, 0
9           .endif
10  .endif
11
12  ;Play 2
13  .elseif eax==VK_Q
14      .if gStatus == 1
15      .if FinDisturb > 0
16              mov inFinSkill, 400
17              ; TODO
18              ;mov inFinSkill2, -30
19              mov FinDisturb, 0
20          .endif
21  .endif
22  .endif
```

然后在 TimerProc 进一步控制新柱子的创建。在创建新柱子的时候依旧会调用自定义的 Random 函数生成一个随机数，如果释放了大招随机数会加上 250 否则加上 195，将此值作为下柱子的 y 坐标，之后用下柱子的纵坐标减去基础开口宽度和大招状态下额外增加的开口宽度 inFinSkill 获得上柱子的 y 坐标。

Listing 4.18: 大招释放过程

```
mov eax, [edx].OBJECT.x
add eax, 200
add ebx, sizeof OBJECT
add edx, sizeof OBJECT
mov [ebx].OBJECT.x, eax
mov [edx].OBJECT.x, eax
mov ecx, inFinSkill
mov [ebx].OBJECT.wide, ecx
mov [edx].OBJECT.wide, ecx

 invoke Random, 160
 .if inFinSkill > 0
 add eax, 250
 .else
 add eax, 195
 .endif
 mov [edx].OBJECT.y, eax
 sub eax, 450
 sub eax, inFinSkill
 mov [ebx].OBJECT.y, eax

mov addedFlag2, 0
```

大招持续过程 10s，额外增加的开口宽度大小 inFinSkill 逐渐减小，伴随着上下柱子开口大小会逐渐减少恢复至正常宽度。

Listing 4.19: 大招效果逐渐减弱至无

```
    .if inFinSkill>0
        dec inFinSkill
    .elseif inFinSkill<0
        inc inFinSkill
    .endif
```