

Assembly Language

貪食蛇遊戲

1092574 王清霖

1092929 徐文獻

1092923 林品安



目錄

01

導論

動機
目的

02

系統概述

背景
系統介紹

03

系統設計

初始設定
函式概述

系統製作

使用者輸入設定
重要使用函數與類別 說明

04

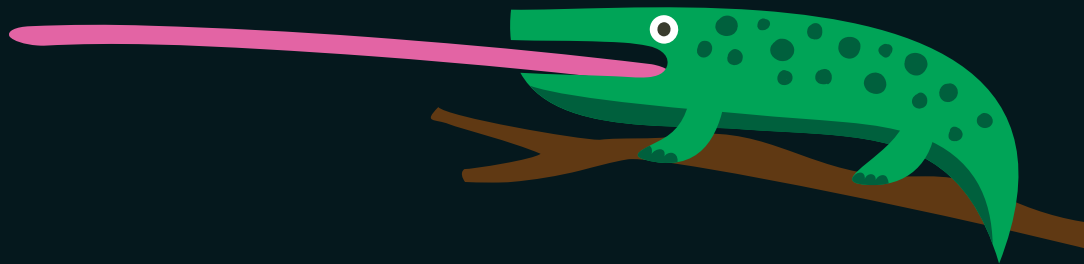
系統評估

達成目標
未達成目標
心得

05

01

導論






動機

靈感是來自於上課所使用的DOSBOX，由黑白兩色形成的單調視窗，使我們想起了歷史久遠的貪食蛇遊戲，其也是一款介面簡單，且易於上手的趣味遊戲。



目的

利用組合語言實作出貪吃蛇遊戲，以最簡單的設計元素，營造出復古的遊玩風格



02

系統概述

2.1 背景

- 貪食蛇遊戲
- 玩家操控蛇往上、下、左、右移動
- 蛇吃到食物時分數增加
- 蛇撞到牆或自己時遊戲結束

2.2 系統簡介

使用工具：

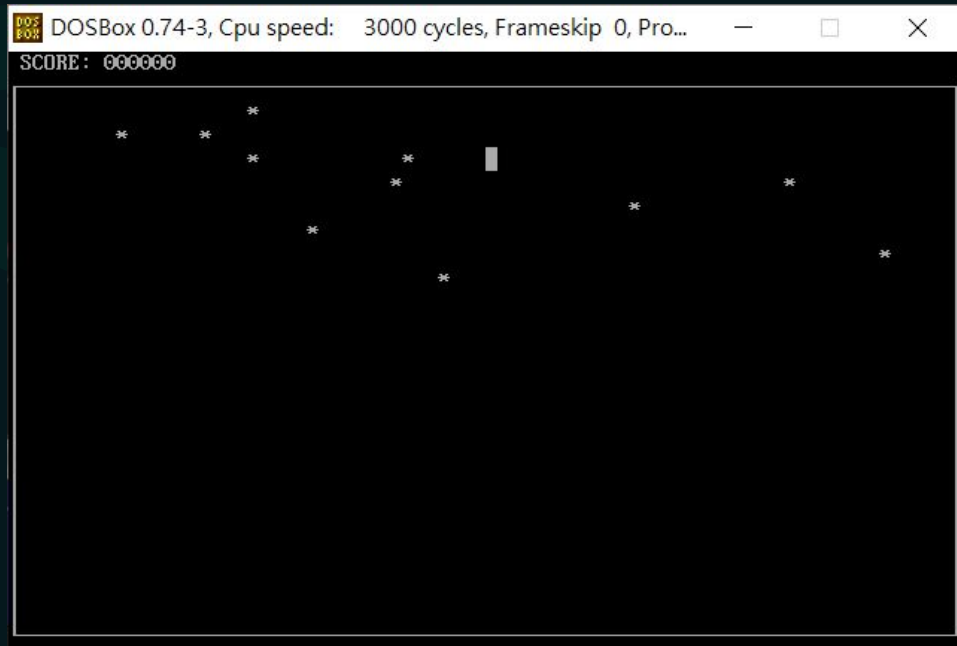
- Vscode - 編輯程式碼
- nasm - 將.asm檔案轉換成.com檔案
- Dosbox - 執行.com檔案

2.2 系統簡介




- 執行介面如左圖
- 按下任意鍵即開始遊戲

2.2 系統簡介



- 方向鍵 => 上、下、左、右移動
- ESC => 跳出遊戲
- 吃到食物 => 分數+1



03



系統設計

3.1 初始設定

- ❑ `.title` : 遊戲一開始進入畫面的動畫顯示
- ❑ `.text_1`, `.text_2`, `.text_3`, `.text_4` : 初始的 home 頁面的顯示畫面
- ❑ `bits 16` : 設置程式的位元模式為16位
- ❑ `org 100h` ; 將程式的起始位址設置為100h
- ❑ `Draw_border`, `buffer_clear`, `create_initial_foods`, `reset` : 初始化蛇的位置、顯示緩衝區、畫面渲染

3.2 函式概述 – 遊戲介面

❑ Hide_cursor : 隱藏滑鼠 (將滑鼠移動到畫面外)

❑ Clear_keyboard_buffer : 清空鍵盤緩衝區 (16h 中斷 BIOS 鍵盤函式)

❑ Exit_process : 結束程式執行

❑ buffer_clear : 清空緩衝區

❑ Buffer_write : 向緩衝區寫入字符 (單個字符, 如身體或食物等)

❑ Buffer_read : 從緩衝區讀取字符

❑ Buffer_print_string : 向緩衝區寫入字符 (一次一長串, 遊戲標題等)

❑ Buffer_render : 將緩衝區內容顯示於螢幕上

❑ Show_title : 顯示遊戲標題 (包括等待使用者按下按鍵)

❑ Print_score : 顯示分數

3.2 函式概述 - 程式邏輯

- ❑ Start , Sleep : 遊戲開始、暫停遊戲
- ❑ update *snake_head* : 依據按鍵更改蛇的頭方向(包括 update_snake_direction)
- ❑ Update_snake_tail : 更新蛇尾位置
- ❑ Check_snake_new_position : 檢查蛇的狀態, 是否死亡或吃到食物
- ❑ Create_initial_foods : 透過 create_food 生成食物, 預設為 10 個食物
- ❑ Reset : 遊戲重新初始化
- ❑ Start_playing : 開始遊戲
- ❑ Show_game_over : 顯示 game over
- ❑ Draw_border : 畫遊戲邊界

04

系統製作



4.1 使用者輸入設定

- 按下任意鍵開始遊戲

```
162 .wait_for_key: ; 等待使用者按下按鍵
163     mov si, .text_4 ; 將 .text_4 的位址載入 si 寄存器
164     mov di, 1388 ; 1388 作為字符串的偏移量
165     call buffer_print_string
166     call buffer_render
167     mov si, 5
168     call sleep
169     mov ah, 1 ; 將值 1 載入 ah 寄存器，準備獲取鍵盤輸入
170     int 16h ; : 觸發鍵盤服務中斷
171     jnz .continue
172     mov si, .text_3
173     mov di, 1388
174     call buffer_print_string
175     call buffer_render
176     mov si, 10
177     call sleep
178     mov ah, 1
179     int 16h
180     jz .wait_for_key
```


4.1 使用者輸入設定

- 四個方向鍵控制蛇向上、下、左、右移動
- ESC鍵控制離開

```
231 update_snake_direction: ;依據按鍵更新蛇的方向
232     mov ah, 1
233     int 16h ;從緩衝區中獲取按鍵
234     jz .end
235     mov ah, 0h ; retrieve key from buffer
236     int 16h
237     cmp al, 27 ; ESC:退出程式
238     jz exit_process
239     ;根據按鍵的不同，將相應的數值存儲在 snake_direction 變量中
240     cmp ah, 48h ; up
241     jz .up
242     cmp ah, 50h ; down
243     jz .down
244     cmp ah, 4bh; left
245     jz .left
246     cmp ah, 4dh; right
247     jz .right
248     jmp update_snake_direction ;使用無條件跳轉回 update_snake_di
```

4.2 重要使用函數與類別說明

```
check_snake_new_position:
    mov ch, 0
    mov cl, [snake_head_x]
    mov dh, 0
    mov dl, [snake_head_y]
    call buffer_read ;使用 buffer_read 函數讀取蛇頭新位置的內容，並將
    cmp bl, 8 ;蛇撞到了邊界或自身的身體，程式碼會設置 is_game_over 變
    jle .set_game_over
    cmp bl, '*' ;蛇吃到了食物，程式碼會增加 score 變量的值，然後呼叫 .
    je .food
    cmp bl, ' ' ;蛇移動到了空白的位置，程式碼會呼叫 update_snake_tail
    je .empty_space
.set_game_over:
    cmp al, 1
    mov byte [is_game_over], al
.write_new_head:
    mov bl, 1
    mov ch, 0
    mov cl, [snake_head_x]
    mov dh, 0
    mov dl, [snake_head_y]
    call buffer_write
    ret
.food:
    inc dword [score]
    call .write_new_head
    call create_food
    jmp .end
.empty_space:
    call update_snake_tail
    call .write_new_head
.end:
    ret
```

- 將蛇頭的新位置存在ch、cl、dh、dl等暫存器中
- 由緩衝器讀取蛇頭的位置
- 如果蛇頭位置等於8代表撞到自己或是撞到邊界(結束遊戲)
- 讀到“*”代表吃到食物，會更新蛇頭與再增加食物
- 讀到空白，更新蛇尾、蛇頭

4.2 重要使用函數與類別說明

```
update_snake_tail: ;更新蛇尾的位置
    ;將蛇尾的前一個位置保存在 snake_tail_previous_x 和 snake_tail_previous_y
    mov al, [snake_tail_y]
    mov byte [snake_tail_previous_y], al
    mov al, [snake_tail_x]
    mov byte [snake_tail_previous_x], al
    mov ch, 0
    mov cl, [snake_tail_x]
    mov dh, 0
    mov dl, [snake_tail_y]
    ;程式碼從緩衝區讀取一個字符，並根據該字符來更新蛇尾的新位置。根據字符的不同，蛇尾
    call buffer_read
    cmp bl, 8 ; up
    jz .up
    cmp bl, 4 ; down
    jz .down
    cmp bl, 2 ; left
    jz .left
    cmp bl, 1 ; right
    jz .right
    jmp exit_process

.up:
    dec word [snake_tail_y]
    jmp .end

.down:
    inc word [snake_tail_y]
    jmp .end

.left:
    dec word [snake_tail_x]
    jmp .end

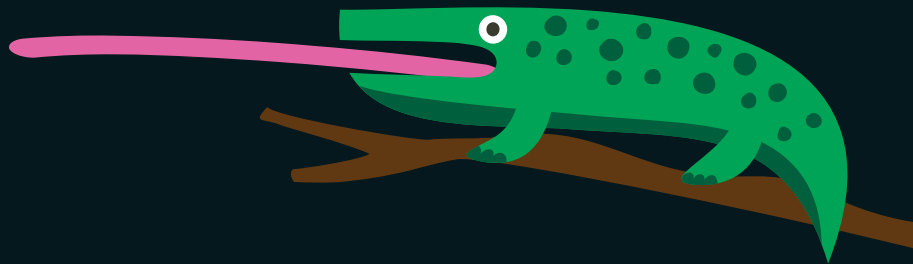
.right:
    inc word [snake_tail_x]
    ;移動完成後，程式碼將蛇尾之前的位置設置為空格字符，以擦除蛇尾之前的痕跡
    .end:
    mov bl, ' '
    mov ch, 0
    mov cl, [snake_tail_previous_x]
    mov dh, 0
    mov dl, [snake_tail_previous_y]
    call buffer_write

ret
```

- 將目前蛇尾的位置保存到 snake_tail_previous_x、snake_tail_previous_y
- 依照蛇的方向移動蛇尾
- 蛇前進後，消除走完後的位置



05



系統評估

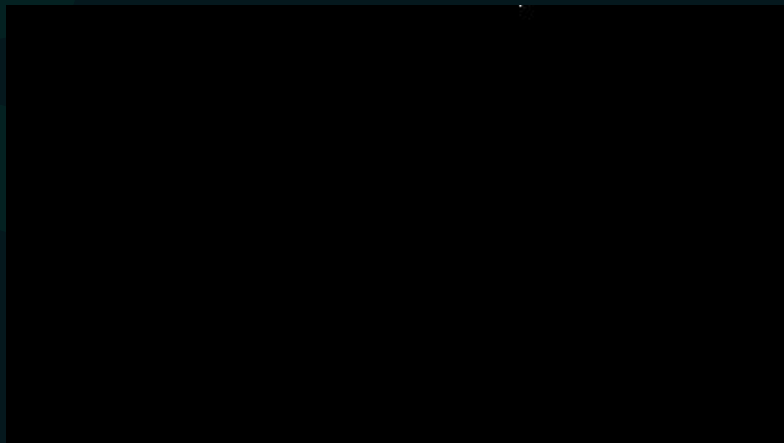


5.1 達成&未達成目標

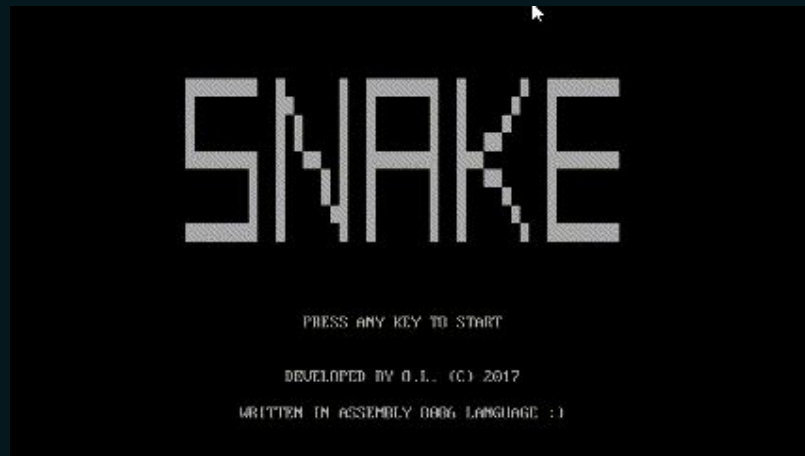
- ✓ 開頭遊戲動畫
- ✓ 鍵盤操作蛇
- ✓ 地圖生成
- ✓ 隨機生成食物
- ✓ 蛇長度更新
- ✓ 計算分數
- ✓ 程式重複執行

- ✗ 分數紀錄
- ✗ 開頭選項操作
- ✗ 音樂

5.2 心得與DEMO



開始畫面



執行畫面

5.2 心得與DEMO

剛開始撰寫組合語言時，高階語言一行就能做到的，換成組合語言竟然要好幾倍的行數，讓我們很不適應，也帶來一定的混亂。

最初構想期中專題也苦惱了一陣子，才選定此次的主題。過程中學到許多組合語言的邏輯與技巧，成功實作出貪吃蛇遊戲，獲益良多

Thanks

