状态管理器实现 （上）– 控制台应用

1 目标

使用 C 语言，Visual Studio 工具，设计并完成基于控制台应用的状态控制器，管理具有 2 个游戏关卡的状态切换，要求能显示正确的状态函数调用顺序（Load，Initialize，Update，Draw，Free，Unload）。

2 要求

Output.txt

建立文件 output.txt，该文件用于状态控制器的输出。

在系统初始化函数中 open 该文件，并在系统退出函数中 close。

GameStateList.h

建立头文件 GameStateList.h，在该头文件中声明枚举类型，包括 Level1，Leve2，Restart，Quit。 （已提供）

GSM

1） GSM 负责状态切换，并调用相应的状态函数。切记：使用函数指针调用状态函数。

2） 当切换到新状态的时候，函数指针需要指向新状态的状态函数。

3） 你需要声明 3 个状态指示器，Previous、Current、Next

4） 你还需要编写两个函数

a) GSM_Initialize，该函数的功能是将 3 个状态指示器全部初始化为初始状态，初始状态作为函数参数传递进来。当该函数执行时，应当行输出“GSM：Initialize”到 output.txt 中

b) GSM_Update，更新 6 个函数指针，使它们指向新状态的状态函数，同时还应当行输出“GSM：Update”到 output.txt 中

5） GSM 代码需要两个文件

a) GameStateManager.c

b) GameStateManager.h

实现两个虚拟关卡 Level1 和 Level2

1） 必须为每一个关卡定义自己的 6 个状态函数，load，initialize，update，draw，free 以及 unload

2） 以上每个函数执行时，需要向 output.txt 行输出，具体如下

a) Level1 的 load 需要输出“Level1:Load”

b) Level1 的 initialize 需要输出“Level1:Initialize”

c) Level1 的 update 需要输出“Level1:Update”

d) Level1 的 draw 需要输出“Level1:Draw”

e) Level1 的 free 需要输出“Level1:Free”

f) Level1 的 unload 需要输出“Level1:Unload”

g) Level2 的 load 需要输出“Level2:Load”

h) Level2 的 initialize 需要输出“Level2:Initialize”

i) Level2 的 update 需要输出“Level2:Update”

j) Level2 的 draw 需要输出“Level2:Draw”

k) Level2 的 free 需要输出“Level2:Free”

l) Level2 的 unload 需要输出“Level2:Unload”

3） 以上函数不允许直接调用，必须通过指向它们的函数指针调用

4） 两个 level 需要 4 个文件

a) Level1.c

b) Level1.h

c) Level2.c

d) Level2.h

System Handler

系统处理器目前只处理虚拟的 GSM，但是以后，会陆续添加输入管理器、图形管理器等。

1） 该功能需要两个函数

a) System_Initialize：打开 output.txt，并行输出“System:Initialize”

b) System_Exit：行输出“System:Exit”到 output.txt

2） 代码编写需要两个文件

a) System.c

b) System.h

Input Handler

目前实现虚拟的 Input Handler，以后会填充这一部分，读取键盘鼠标输入。

1） 该功能需要一个函数

a) Input_Handle：行输出“Input_Handle”到 output.txt

2） 代码编写需要两个文件

a) Input.c（已提供）

b) Input.h（已提供）

main

在 main 函数里实现完整的 game flow。

该 game flow 需要包含 game loop。

需要完成 1 个文件，main.c

3 状态描述

Level1

1） 在 Level1 里定义一个整型变量 Level1_Counter，在 level1 的 load 函数里初始化该变量

2） Level1_Counter 的初始值来源于外部文件 Level1_Counter.txt（已提供）

3） 在 update 函数里更新该变量，令 Level1_Counter 减 1

4） 当 Level1_Counter 等于 0，切换到 level2

Level2

1） 在 Level2 里定义两个整型变量，Level2_Counter 和 Level2_Lives

2） Level2_Counter 要在 level2 的 Initialize 函数中初始化，初始值来源于外部文件 Level2_Counter.txt（已提供）

3） Level2_lives 在 level2 的 load 函数里初始化，初始值来源于外部文件 Level2_Lives.txt（已提供）

4） 在 update 函数里更新该变量，令 Level2_Counter 减 1

5） 当 Level2_Counter 等于 0，Level2_lives 减 1

a) 如果 Level2_lives 等于 0，退出游戏

b) 如果 Level2_lives 大于 0，restart level2

程序清单

GameStateList.h（已提供）

GameStateManager.c

GameStateManager.h

Level1.c

Level1.h

Level2.c

Level2.h

System.c

System.h

Input.c（已提供）

Input.h（已提供）

main.c

测试

Level1_Counter 的初始值为 3，Level2_Counter 的初始值为 2，Level2_lives 的初始值为 2
运行程序后，output.txt 应该如下：

System:Initialize
GSM:Initialize
GSM:Update
Level1:Load
Level1:Initialize
Input:Handle
Level1:Update

Level1:Draw
Input:Handle
Level1:Update
Level1:Draw
Input:Handle
Level1:Update
Level1:Draw
Level1:Free
Level1:Unload
GSM:Update
Level2:Load
Level2:Initialize
Input:Handle
Level2:Update
Level2:Draw
Input:Handle
Level2:Update
Level2:Draw
Level2:Free
Level2:Initialize
Input:Handle
Level2:Update
Level2:Draw
Input:Handle
Level2:Update
Level2:Draw
Level2:Free
Level2:Unload
System:Exit