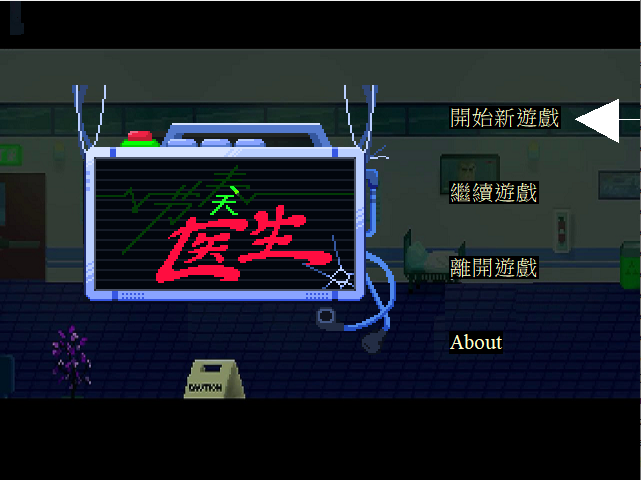
國立臺北科技大學

2021 Spring 資工系物件導向程式實習

期末報告

節奏醫生(Rhythm Doctor)



第37組

108590036 沈宗毅

108590044 何柏憲

**目錄**

**一、** **簡介** 1

1. 動機 1

2. 分工 1

**二、** **遊戲介紹** 2

1. 遊戲說明 2

2. 遊戲圖形 3

3. 遊戲音效 6

**三、** **程式設計** 7

1. 程式架構 7

2. 程式類別 8

3. 程式技術 8

**四、 結語** 9

1. 問題及解決方法 9

2. 時間表 11

3. 貢獻比例 12

4. 自我檢核表 12

5. 收穫 12

6. 心得 13

**五、 附錄** 14

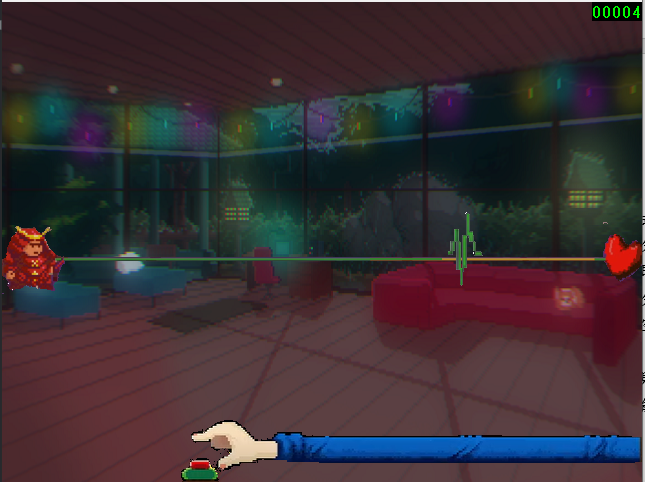
1. **簡介**
2. 動機

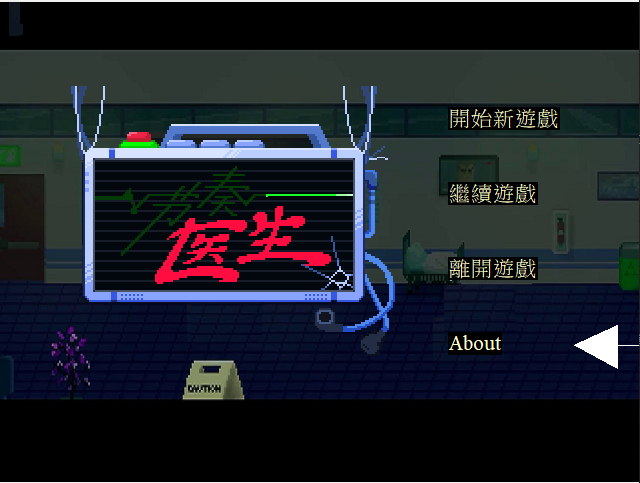
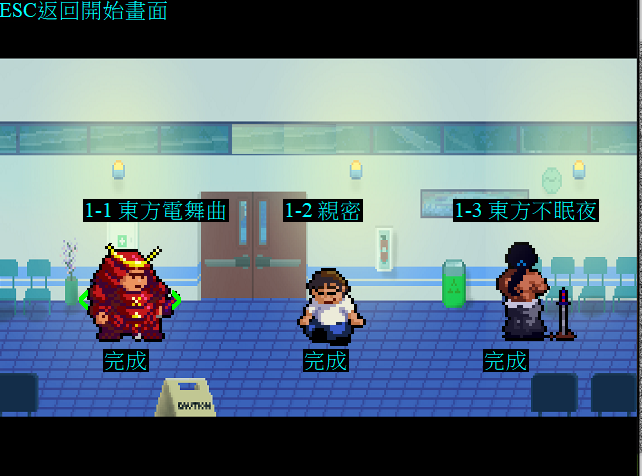
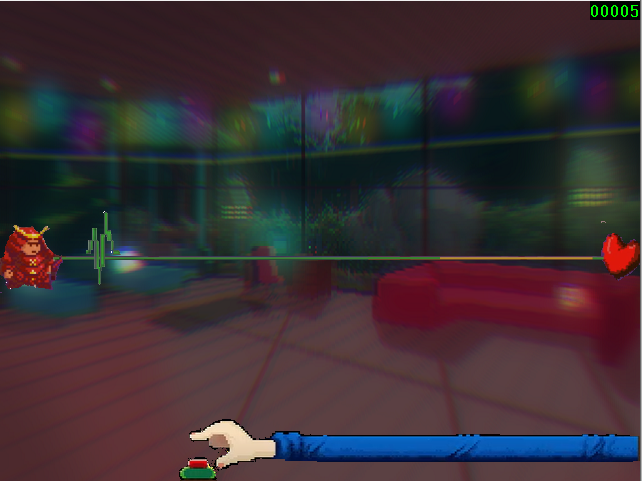
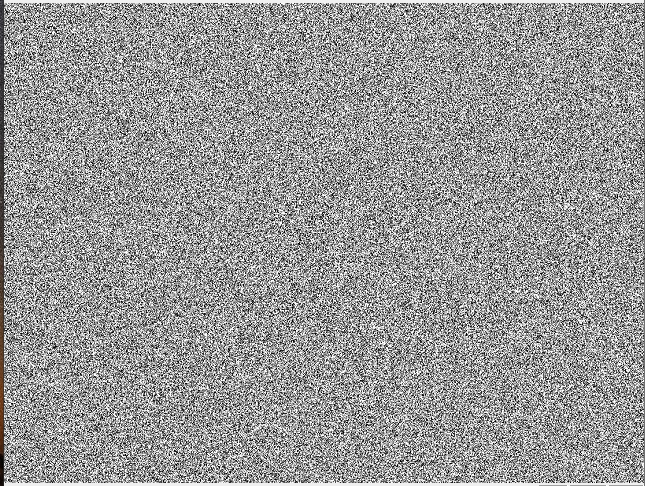
　　學期初的時候，聽到物件導向設計實習需要找一個遊戲來複製，我們第一個想到的就是節奏醫生，那時候節奏醫生的正式版正好剛出沒多久，看到遊戲裡面的各種特效，就讓人躍躍欲試，迫不及待的看能不能重現遊戲中各種繽紛的特效。另一方面，想到音樂遊戲就會想到需要多個按鍵去對應到各種節拍，不過節奏醫生只需要一個按鍵，實際上去設計按鍵就會比一般音樂遊戲更少。

1. 分工

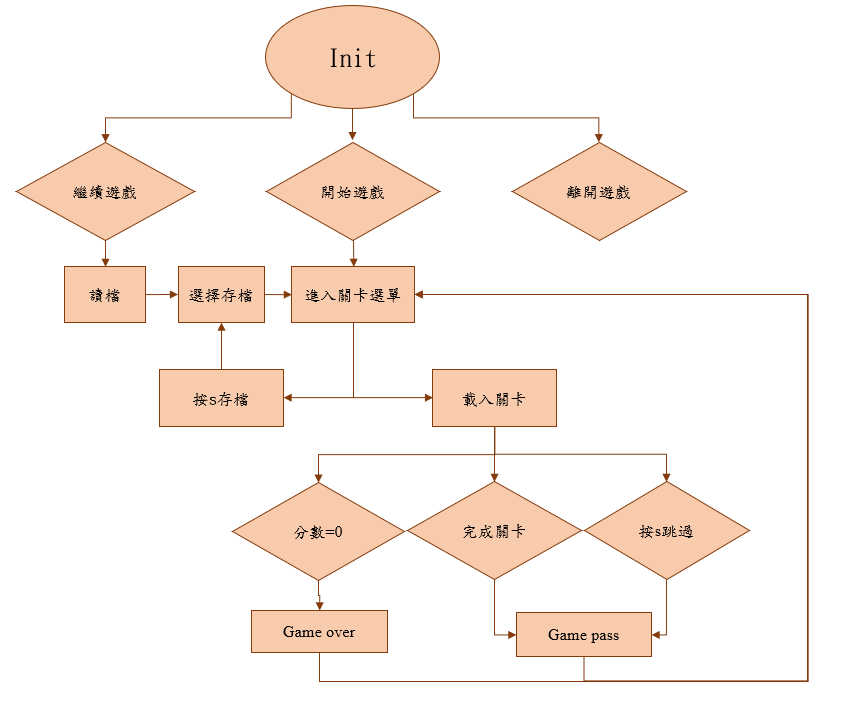
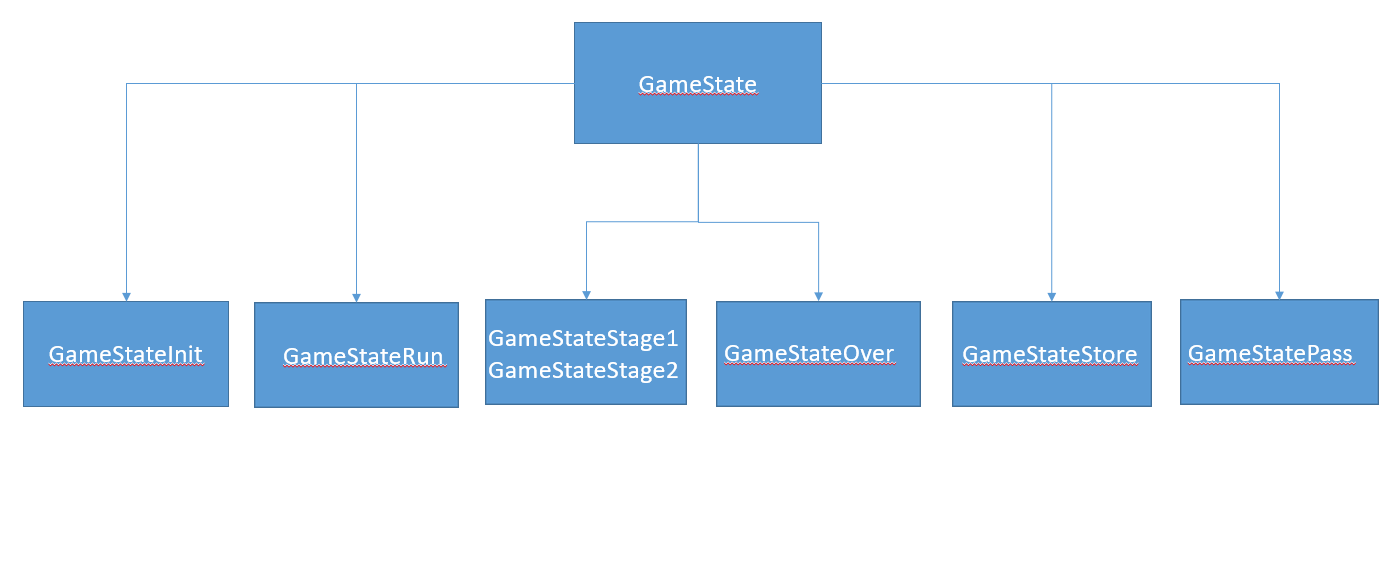
|  |  |
| --- | --- |
| 何柏憲 | 負責大部分的介面與動畫顯示和一部分的遊戲功能。 |
| 沈宗毅 | 負責大部分的遊戲功能。 |

1. **遊戲介紹**
   1. 遊戲說明
      1. 遊玩方式  
         音樂部分：隨著音樂節奏會特別有一個重音拍（音效）提示，即可按下空白鍵。

動畫部分：中間會有心電圖圖示，當顯示到黃色區段即可按下空白鍵。如下圖所示

* + 1. 遊戲規則  
       按下空白鍵後，有對應到拍子和心電圖就會加分，否則扣分，分數會有預設值，並且在遊戲中會顯示於右上角。分數歸零則遊戲結束。如順利完成，遊戲結束後的評價由當前關卡的分數占總節拍比來決定。
    2. 遊戲動畫  
       各個關卡中有各種不同的動畫，還有一些干擾。
    3. 特殊功能  
       我們有實作存檔功能，開始新遊戲進入選關卡的界面按s即可存檔，有三個欄位可供玩家做存檔選擇，如欲刪除，直接覆蓋即可。除此之外，未完成前一關時下一關將不會解鎖。
    4. 密技  
       按下S鍵即可跳關，並以最高評價S完成關卡。
  1. 遊戲圖形  
        
       
     
  2. 遊戲音效

|  |  |
| --- | --- |
| 音樂檔名 | 說明 |
| 1.mp3 | 第一關音樂 |
| 2.mp3 | 第二關音樂 |
| 3.mp3 | 第三關音樂 |
| 4.mp3 | 第四關音樂 |
| 5.mp3 | 第五關音樂 |
| 6.mp3 | 第六關音樂 |
| dingT1.mp3 | 切換所選選單音效 |
| click.mp3 | 遊戲按空白鍵音效 |
| menu.mp3 | 主選單畫面音樂 |

1. **程式設計**
   1. 程式架構
   2. 程式類別

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| ClongGray | 23 | 42 | 類似於CBall的class然後可以控制圖形是否顯示。 |
| mygame | 254 | 1782 | 兩個遊戲選單、存檔畫面、遊戲主要運作的程式碼和過關畫面。 |
| stage1 | 12 | 0 | 第一關節奏資料 |
| stage2 | 12 | 0 | 第二關節奏資料 |
| stage3 | 12 | 0 | 第三關節奏資料 |
| stage4 | 12 | 0 | 第四關節奏資料 |
| stage5 | 12 | 0 | 第五關節奏資料 |
| stage6 | 12 | 0 | 第六關節奏資料 |
|  |  |  |  |

* 1. 程式技術

　　我們有使用到vector，vector的表現一如資料結構中的陣列，又有很多額外的功能。

1. 結語
2. 問題及解決方法
3. 按空白鍵的效果:

按空白鍵後會出現遊戲特有的長條，但他只會出現在那一瞬間，雖然現在想想其實蠻簡單的，但那時沒想到可以用一個布林值去控制他。

1. 控制時間點:

因為這遊戲是節奏遊戲，所以我們要好好掌握什麼時間點按空白鍵會加分，我們有使用ctime的clock去幫助我們判斷時間點，這邊沒遇到什麼太多的阻礙。

1. 讀寫檔:

我們用到之前學過的讀檔技巧幫助我們做存讀檔的動作，主要就是判斷什麼時候存檔，什麼時候讀檔，然後把一些關鍵值放到記事本裡面，當我們要讀檔時，我們把那些關鍵值取出來套用到遊戲上。

1. 關卡設定:

我們用level來判斷目前要進入的關卡，好讓GameStateRun判斷現在的level等級載入相對應的關卡資訊，共同的關卡資訊用不同的stage.h存著，MaxLevel表示目前最多可以玩到什麼等級，以此來判斷哪些是通過關卡，哪些未解鎖，或還沒去完成的關卡，隨著通關數增加MaxLevel也會慢慢上升。

1. 拍子顯示:

完成一些基本遊戲機制就是顯示問題，剛開始是有音樂，可是畫面完全沒有參考價值，這時候就要完善可以讓玩家參考的畫面，可是這遊戲一首歌拍子很多，不太可能把每個拍子的時間點還有位置抓出來，所以我們觀察了一下節奏，第一章每拍的時間都很平均，我們先把第一拍跟最後一拍的時間點抓出來，大部分的時候都是7拍，所以7等分切割，這樣就可以得到1~7拍的時間，很多時候位置也是1 2 3 4 5 6 7這樣的位置，這樣就可以獲得每個時間點跟位置，當然前兩關比較單純，是這樣，但1-x就不同了，他很多時候顯示的位置不是1 2 3 4 5 6 7那麼單純，可能是1 3 1 2 3 2 4 6 7，而且也不是只有7拍，這關我們就花很久，因為他變化很多，所以我們真的是把每個時間點的變化都抓出來然後特別調整，第2章的節奏跟1不太一樣，那是1顯示一陣子然後很快到7，有些是1後馬上到7，這邊就是用迴圈去分割然後控制拍子在每個位置的時間比例，這個環節準備了location陣列存取每個拍子在甚麼時間點出現的位置是什麼，everytime存取分割出來的時間好做後續判斷。

1. 拍子抖動:

一開始拍子是不會跳動的，換位置也不會跳動，就看起來很不生動，但我們用了一個簡單的jump方法控制他jump，然後用jump\_time\_list紀錄什麼位置要跳動幾次，像1-x有些拍子可能會跳動兩次或三次，有的還會四次，所以才要用jump\_time\_list去存每個拍子在什麼時間點的跳動次數。

1. 時間表

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 組員-何柏憲(小時) | 組員-沈宗毅(小時) | 說明 |
| 1 | 1 | 1 | 練習git上傳、tutorial |
| 2 | 3 | 3 | 練習git上傳、tutorial |
| 3 | 8 | 8 | 找素材、選單畫面設計 |
| 4 | 6 | 6 | 選單功能、音樂與畫面優化 |
| 5 | 6 | 6 | 做出第一版節拍判定、 遊戲中物件的顯示及音效、選單音效 |
| 6 | 2 | 2 | 確定第一關的節拍點 |
| 7 | 3 | 3 | 做出第二版節奏判定 |
| 8 | 6 | 6 | 遊戲關卡畫面設計 |
| 9 | 2 | 2 | 篩選歌曲 |
| 10 | 6 | 6 | 做出第一關的雛形 |
| 11 | 9 | 9 | 選擇關卡畫面、功能與動畫 |
| 12 | 6 | 6 | 存檔畫面及功能 |
| 13 | 9 | 9 | 新增第二關、第三關 |
| 14 | 10 | 10 | 優化選擇關卡畫面 |
| 15 | 10 | 10 | 新增第四關、研發不同拍子的心電圖顯示 |
| 16 | 10 | 10 | 關卡畫面優化、套用心電圖顯示 |
| 17 | 20 | 20 | 新增兩個關卡、完成及優化六個關卡功能及特效 |

1. 貢獻比例

沈宗毅：50%、何柏憲：50%

1. 自我檢核表

|  |  |  |  |
| --- | --- | --- | --- |
| 項目 | 項目 | 完成否 | 無法完成原因 |
| 1 | 解決 Memory leak | ■已完成 |  |
| 2 | 自定遊戲 Icon | ■已完成 |  |
| 3 | 全螢幕啟動 | ■已完成 |  |
| 4 | 有 About 畫面 | ■已完成 |  |
| 5 | 初始畫面說明按鍵及滑鼠之用法與密技 | ■已完成 |  |
| 7 | 上傳 setup/apk/source 檔 | ■已完成 |  |
| 8 | setup 檔可正確執行 | ■已完成 |  |
| 9 | 報告字型、點數、對齊、行距、頁碼等格式正確 | ■已完成 |  |

1. 收穫

沈宗毅：更了解vector的用途，因為我們是節奏遊戲，要用到很多vector去紀錄各個節奏點，還有各個節奏的行為， 雖然vector很好用，但還是有很多事情要注意，其中最常遇到的就是vector subscript out of range，然後我就知道 不能直接賦值，還有一些邊界問題的錯誤也比較不會犯了。 除錯技巧大概會抓到底是哪塊出了問題，可能會先把一部份註解掉或是用其他方式代替，慢慢找出有問題的代碼。

何柏憲：在本次的實習中，我了解到遊戲中動畫製作相當的不容易，就算我們製作的只是2d的遊戲，圖形還是會非常的複雜，尤其是我們用這個框架下去製作，絕對不會比一般的遊戲引擎還要輕鬆，可能要去計算一些拍子上的顯示，還有一些不規則的拍子，會用到許多數學上的概念，在經過多次的微調後大概就可以讓音樂和畫面顯示同步。最主要判定的東西有用到time.h，來記錄時間的節奏，我們把紀錄時間的節奏在寫入檔案中就有用到ofstream，然後就使用這個檔案下去作為節拍的基準。

1. 心得

沈宗毅：這次oop實習是我第一次做的一個對我來說規模算大的專案，也讓我更了解一些遊戲設計者的心思，從一開始什麼 想法都沒有到，然後慢慢做，開始有越來越多想法，越來越了解程式架構，慢慢實踐一些心中的想法， 雖然我覺得這次沒有做到非常好，但這個過程讓我對物件導向程式設計的概念有很大的提升。

何柏憲：這次物件導向程式設計實習是我上大學以來花最多時間的專案，但實際上成果不甚滿意，覺得自己必須要積極一點，不然以後會在各種方面都很吃虧。不過經由這次的實習，真的學習到很多，起初一開始完全不知道怎麼架構整個遊戲，藉由一週一週的下去討論和分工，到期中時遊戲已經有基本的雛型，就覺得原來做遊戲就是這樣，不是一次直接完成，而是慢慢地開發，慢慢地增加東西，不要想著要做得多好，而是自己可以做到什麼，按部就班，就不會迷失方向。

1. **附錄**

mygame.cpp

/\*

\* mygame.cpp: 本檔案儲遊戲本身的class的implementation

\* Copyright (C) 2002-2008 Woei-Kae Chen <wkc@csie.ntut.edu.tw>

\*

\* This file is part of game, a free game development framework for windows.

\*

\* game is free software; you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation; either version 2 of the License, or

\* (at your option) any later version.

\*

\* game is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

\*

\* History:

\* 2002-03-04 V3.1

\* Add codes to demostrate the use of CMovingBitmap::ShowBitmap(CMovingBitmap &).

\* 2004-03-02 V4.0

\* 1. Add CGameStateInit, CGameStateRun, and CGameStateOver to

\* demonstrate the use of states.

\* 2. Demo the use of CInteger in CGameStateRun.

\* 2005-09-13

\* Rewrite the codes for CBall and CEraser.

\* 2005-09-20 V4.2Beta1.

\* 2005-09-29 V4.2Beta2.

\* 1. Add codes to display IDC\_GAMECURSOR in GameStateRun.

\* 2006-02-08 V4.2

\* 1. Revise sample screens to display in English only.

\* 2. Add code in CGameStateInit to demo the use of PostQuitMessage().

\* 3. Rename OnInitialUpdate() -> OnInit().

\* 4. Fix the bug that OnBeginState() of GameStateInit is not called.

\* 5. Replace AUDIO\_CANYON as AUDIO\_NTUT.

\* 6. Add help bitmap to CGameStateRun.

\* 2006-09-09 V4.3

\* 1. Rename Move() and Show() as OnMove and OnShow() to emphasize that they are

\* event driven.

\* 2006-12-30

\* 1. Bug fix: fix a memory leak problem by replacing PostQuitMessage(0) as

\* PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE,0,0).

\* 2008-02-15 V4.4

\* 1. Add namespace game\_framework.

\* 2. Replace the demonstration of animation as a new bouncing ball.

\* 3. Use ShowInitProgress(percent) to display loading progress.

\* 2010-03-23 V4.6

\* 1. Demo MP3 support: use lake.mp3 to replace lake.wav.

\*/

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

#include <cstdlib>

#include <time.h>

#include <ctime>

#include <iostream>

#include <vector>

#include <fstream>

bool times = true;

int start,END =0;

int getscore; //獲得分數

int level; //level選擇遊戲的難度 MaxLevel破解的最高難度

int MaxLevel;

bool write\_load = false; //true是load false是write

namespace game\_framework {

CGameStateInit::CGameStateInit(CGame\* g)

: CGameState(g)

{

}

CPractice::CPractice(){

//起始位置

x = 85; //x end = 470

y = 220;

}

int k,i = 0;

void CPractice::OnMove(int position) {

//移動行為

x = 85;

for (int i = 1; i < position; i++) {

x += 60;

}

if (x >= 470) {

x = 85;

}

}

void CPractice::OnJump() {

//跳動行為

if (y == 220) {

y = 210;

}

else

y = 220;

}

void CPractice::OnJump2() {

if (y == 220) {

y = 400;

}

else

y = 220;

}

void CPractice::LoadBitmap() {

pic.LoadBitmap(IDB\_BITMAP42,RGB(255,255,255));

}

void CPractice::OnShow() {

pic.SetTopLeft(x, y);

pic.ShowBitmap();

}

int CPractice::getX() {

return this->x;

}

void CBouncingBall::SetXY(int x, int y) {

this->x = x;

this->y = y;

}

void CBouncingBall::SetFloor(int floor) {

this->floor = floor;

}

void CBouncingBall::SetVelocity(int velocity) {

this->velocity = velocity;

this->initial\_velocity = velocity;

}

void CGameStateInit::OnInit()

{

//

// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人

// 等的不耐煩，遊戲會出現「Loading ...」，顯示Loading的進度。

//

ShowInitProgress(0); // 一開始的loading進度為0%

//

// 開始載入資料

//

for (int i = IDB\_BITMAP9; i <= IDB\_BITMAP16; i++) {

title1.AddBitmap(i,RGB(255,255,255));

}

title1.SetTopLeft(40, 80);

about.LoadBitmapA(IDB\_BITMAP75);

x = 450;

y = 70;

title1.SetDelayCount(2);

BG.LoadBitmap(IDB\_BITMAP43);

logo.AddBitmap(IDB\_INITSELECTBOX,RGB(0,0,0));

logo.AddBitmap(IDB\_BITMAP44,RGB(0, 0, 0));

logo.AddBitmap(IDB\_BITMAP45,RGB(0, 0, 0));

logo.SetTopLeft(x, 70);

CAudio::Instance()->Load(AUDIO\_DING, "sounds\\click.mp3");

CAudio::Instance()->Load(AUDIO\_CLICK, "sounds\\dingT1.mp3");

about.SetIsShow(false);

}

void CGameStateInit::OnBeginState()

{

if (times) {

CAudio::Instance()->Load(AUDIO\_LAKE, "sounds\\menu.mp3");

times = false;

}

CAudio::Instance()->Play(AUDIO\_LAKE, true);

}

void CGameStateInit::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_ESC = 27;

const char KEY\_SPACE = ' ';

const char KEY\_ENTER = 13;

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

//const char KEY\_ENTER = 0;

if (nChar == KEY\_ENTER) {

if (y == 70) {

level = 1;

CAudio::Instance()->Play(AUDIO\_DING);

MaxLevel = 0;

GotoGameState(GAME\_STATE\_STAGE);// 切換至GAME\_STATE\_STAGE 選關

END = clock();

}

if (y == 145) {

CAudio::Instance()->Play(AUDIO\_DING);

GotoGameState(GAME\_STATE\_STORE);// 切換至GAME\_STATE\_STORE 讀檔畫面

END = clock();

}

else if (y == 220) {

PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE, 0, 0); // 關閉遊戲

}

if (y == 295) {

about.SetIsShow(true);

}

else about.SetIsShow(false);

}

else if (nChar == KEY\_DOWN) {

CAudio::Instance()->Play(AUDIO\_DING);

if (y < 295)

y += 75;

logo.SetTopLeft(450, y);

}

else if (nChar == KEY\_UP) {

CAudio::Instance()->Play(AUDIO\_DING);

if (y > 105)

y -= 75;

logo.SetTopLeft(450, y);

}

if (nChar == KEY\_ESC && about.IsShow())

about.SetIsShow(false);

else if (nChar == KEY\_ESC)

PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE, 0, 0); // 關閉遊戲

}

void CGameStateInit::OnLButtonDown(UINT nFlags, CPoint point)

{

}

void CGameStateInit::OnMove() {

title1.OnMove();

logo.OnMove();

}

void CGameStateInit::OnShow()

{

BG.ShowBitmap();

logo.OnShow();

title1.SetDelayCount(3);

title1.OnShow();

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 200));

pDC->TextOut(450, 105, "開始新遊戲");

pDC->TextOut(450, 180, "繼續遊戲");

pDC->TextOut(450, 255, "離開遊戲");

pDC->TextOut(450, 330, "About");

pDC->SelectObject(fp); // 放掉 font f

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

if (about.IsShow()) {

about.OnShow();

}

about.SetIsAlive(true);

about.SetXY(80, 60);

}

CGameStateStage::CGameStateStage(CGame\* g)

: CGameState(g)

{

}

void CGameStateStage::OnInit()

{

//載入圖片

bg.LoadBitmap(IDB\_BITMAP46);

select.LoadBitmap(IDB\_BITMAP33,RGB(255,255,255));

stage11.AddBitmap(IDB\_BITMAP32, RGB(255, 255, 255));

stage11.AddBitmap(IDB\_BITMAP40, RGB(255, 255, 255));

stage12.AddBitmap(IDB\_BITMAP34, RGB(255, 255, 255));

stage12.AddBitmap(IDB\_BITMAP47, RGB(255, 255, 255));

stage13.AddBitmap(IDB\_BITMAP35, RGB(255, 255, 255));

stage13.AddBitmap(IDB\_BITMAP48, RGB(255, 255, 255));

x = 80;

y = 250;

}

void CGameStateStage::OnBeginState()

{

if (times) {

times = false;

}

CAudio::Instance()->Play(AUDIO\_LAKE);

select.SetTopLeft(x, y);

stage11.SetTopLeft(85, 240);

stage12.SetTopLeft(280, 260);

stage13.SetTopLeft(485, 240);

}

void CGameStateStage::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_ESC = 27; //esc鍵

const char KEY\_SPACE = ' '; //空白鍵

const char KEY\_ENTER = 13; //ENTER鍵

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

const char save = 83; //存檔按鍵S

//藉由位置 判斷level 進入各個關卡

if (nChar == KEY\_ENTER) {

if (x == 80) {

level = 1;

CAudio::Instance()->Play(AUDIO\_DING);

CAudio::Instance()->Stop(AUDIO\_LAKE);

GotoGameState(GAME\_STATE\_RUN);// 切換至GAME\_STATE\_RUN

END = clock();

}

else if (x == 280 && MaxLevel >= 1) {

level = 2;

CAudio::Instance()->Play(AUDIO\_DING);

CAudio::Instance()->Stop(AUDIO\_LAKE);

GotoGameState(GAME\_STATE\_RUN);// 切換至GAME\_STATE\_RUN

END = clock();

}

else if (x == 480 && MaxLevel >= 2) {

level = 3;

CAudio::Instance()->Play(AUDIO\_DING);

CAudio::Instance()->Stop(AUDIO\_LAKE);

GotoGameState(GAME\_STATE\_RUN);// 切換至GAME\_STATE\_RUN

END = clock();

}

}

//向左向右的行為

else if (nChar == KEY\_LEFT) {

if (x > 80)

x -= 200;

select.SetTopLeft(x, y);

}

else if (nChar == KEY\_RIGHT) {

if (MaxLevel >= 3 && 480 <=x) {

GotoGameState(GAME\_STATE\_STAGE2); //另一個選關畫面

}

else if (x < 480) {

x += 200;

select.SetTopLeft(x, y);

}

}

else if (nChar == KEY\_ESC)

GotoGameState(GAME\_STATE\_INIT);// 切換至GAME\_STATE\_INIT

else if (nChar == save) {

write\_load = true;

GotoGameState(GAME\_STATE\_STORE); //存檔

}

}

void CGameStateStage::OnLButtonDown(UINT nFlags, CPoint point)

{

}

void CGameStateStage::OnMove() {

if (x == 80) {

stage11.SetDelayCount(10);

stage11.OnMove();

stage12.Reset();

}

else if (x == 280) {

stage12.SetDelayCount(10);

stage12.OnMove();

stage11.Reset();

stage13.Reset();

}

else if (x == 480) {

stage13.SetDelayCount(10);

stage13.OnMove();

stage12.Reset();

}

}

void CGameStateStage::OnShow()

{

bg.ShowBitmap();

//根據目前的MaxLevel來設定各關卡下面顯示什麼

string stage\_state[3];

if (MaxLevel > 3) {

for (i = 0; i < 3; i++) {

stage\_state[i] = "完成";

}

}

else {

for (i = 0; i < MaxLevel; i++) {

stage\_state[i] = "完成";

}

if (MaxLevel != 3) {

stage\_state[MaxLevel] = "未完成";

for (i = MaxLevel + 1; i < 3; i++) {

stage\_state[i] = "未解鎖";

}

}

}

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fpq;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fpq = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(TRANSPARENT);

pDC->SetTextColor(RGB(0, 255, 255));

pDC->TextOut(0, 0, "ESC返回開始畫面");

pDC->TextOut(85, 200, "1-1 東方電舞曲");

pDC->TextOut(285, 200, "1-2 親密");

pDC->TextOut(455, 200, "1-3 東方不眠夜");

pDC->TextOut(105, 350, stage\_state[0].c\_str());

pDC->TextOut(305, 350, stage\_state[1].c\_str());

pDC->TextOut(485, 350, stage\_state[2].c\_str());

pDC->SelectObject(fpq); // 放掉 font f

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

stage11.OnShow();

stage12.OnShow();

stage13.OnShow();

select.ShowBitmap();

}

//與stage1大同小異

CGameStateStage2::CGameStateStage2(CGame\* g)

: CGameState(g)

{

}

void CGameStateStage2::OnInit()

{

bg.LoadBitmap(IDB\_BITMAP46);

select.LoadBitmap(IDB\_BITMAP33, RGB(255, 255, 255));

stage11.LoadBitmap(IDB\_BITMAP72, RGB(255, 255, 255));

stage12.LoadBitmap(IDB\_BITMAP73, RGB(255, 255, 255));

stage13.LoadBitmap(IDB\_BITMAP74, RGB(255, 255, 255));

x = 80;

y = 250;

}

void CGameStateStage2::OnBeginState()

{

CAudio::Instance()->Play(AUDIO\_LAKE);

if (times) {

times = false;

}

select.SetTopLeft(x, y);

stage11.SetTopLeft(90, 250);

stage12.SetTopLeft(285, 240);

stage13.SetTopLeft(475, 240);

}

void CGameStateStage2::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_ESC = 27;

const char KEY\_SPACE = ' ';

const char KEY\_ENTER = 13;

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

const char save = 83;

if (nChar == KEY\_ENTER) {

if (x == 80 && MaxLevel >= 3) {

level = 4;

CAudio::Instance()->Play(AUDIO\_DING);

CAudio::Instance()->Stop(AUDIO\_LAKE);

GotoGameState(GAME\_STATE\_RUN);// 切換至GAME\_STATE\_RUN

END = clock();

}

else if (x == 280 && MaxLevel >= 4) {

level = 5;

CAudio::Instance()->Play(AUDIO\_DING);

CAudio::Instance()->Stop(AUDIO\_LAKE);

GotoGameState(GAME\_STATE\_RUN);// 切換至GAME\_STATE\_RUN

END = clock();

}

else if (x == 480 && MaxLevel >= 5) {

level = 6;

CAudio::Instance()->Play(AUDIO\_DING);

CAudio::Instance()->Stop(AUDIO\_LAKE);

GotoGameState(GAME\_STATE\_RUN);// 切換至GAME\_STATE\_RUN

END = clock();

}

}

else if (nChar == KEY\_LEFT) {

if (x <= 80) {

GotoGameState(GAME\_STATE\_STAGE);

}

else if (x > 80) {

x -= 200;

select.SetTopLeft(x, y);

}

}

else if (nChar == KEY\_RIGHT) {

if (x < 480)

x += 200;

select.SetTopLeft(x, y);

}

else if (nChar == KEY\_ESC)

GotoGameState(GAME\_STATE\_INIT);// 切換至GAME\_STATE\_RUN

else if (nChar == save) {

write\_load = true;

GotoGameState(GAME\_STATE\_STORE);

}

}

void CGameStateStage2::OnLButtonDown(UINT nFlags, CPoint point)

{

}

void CGameStateStage2::OnMove() {

if (x == 80) {

}

else if (x == 280) {

}

else if (x == 480) {

}

}

void CGameStateStage2::OnShow()

{

bg.ShowBitmap();

string stage\_state[3];

int M2 = MaxLevel - 3;

for (i = 0; i < M2; i++) {

stage\_state[i] = "完成";

}

if (M2 != 3) {

stage\_state[M2] = "未完成";

for (i = M2 + 1; i < 3; i++) {

stage\_state[i] = "未解鎖";

}

}

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fpq;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fpq = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(TRANSPARENT);

pDC->SetTextColor(RGB(0, 255, 255));

pDC->TextOut(0, 0, "ESC返回開始畫面");

pDC->TextOut(85, 200, "2-1");

pDC->TextOut(285, 200, "2-2");

pDC->TextOut(455, 200, "2-3");

pDC->TextOut(105, 350, stage\_state[0].c\_str());

pDC->TextOut(305, 350, stage\_state[1].c\_str());

pDC->TextOut(485, 350, stage\_state[2].c\_str());

pDC->SelectObject(fpq); // 放掉 font f

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

stage11.ShowBitmap();

stage12.ShowBitmap();

stage13.ShowBitmap();

select.ShowBitmap();

}

CGameStateStore::CGameStateStore(CGame\* g)

: CGameState(g)

{

}

void CGameStateStore::OnInit()

{

bg.LoadBitmap(IDB\_BITMAP43);

store1.LoadBitmap(IDB\_BITMAP36, RGB(255, 0, 255));

store2.LoadBitmap(IDB\_BITMAP37, RGB(255, 0, 255));

store3.LoadBitmap(IDB\_BITMAP38, RGB(255, 10, 255));

select.LoadBitmap(IDB\_BITMAP41, RGB(255,255,255));

x = 85;

y = 300;

}

void CGameStateStore::OnBeginState()

{

if (times) {

times = false;

}

select.SetTopLeft(x, y);

store1.SetTopLeft(0, 100);

store2.SetTopLeft(215, 100);

store3.SetTopLeft(430, 100);

}

void CGameStateStore::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_ESC = 27;

const char KEY\_SPACE = ' ';

const char KEY\_ENTER = 13;

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

if (nChar == KEY\_ENTER) {

//判斷存檔位置 寫入或讀取不同記事本

if (!write\_load) {

if (x == 85) {

ifstream ofs("store1.txt");

string l;

getline(ofs, l);

MaxLevel = atoi(l.c\_str());

ofs.close();

GotoGameState(GAME\_STATE\_STAGE);// 切換至GAME\_STATE\_STAGE

}

else if (x == 285) {

ifstream ofs("store2.txt");

string l;

getline(ofs, l);

MaxLevel = atoi(l.c\_str());

ofs.close();

GotoGameState(GAME\_STATE\_STAGE);// 切換至GAME\_STATE\_STAGE

}

else {

ifstream ofs("store3.txt");

string l;

getline(ofs, l);

MaxLevel = atoi(l.c\_str());

ofs.close();

GotoGameState(GAME\_STATE\_STAGE);// 切換至GAME\_STATE\_STAGE

}

}

else {

if (x == 85) {

ofstream ofs;

ofs.open("store1.txt");

ofs.ios\_base::trunc;

ofs << MaxLevel << endl;

ofs.close();

GotoGameState(GAME\_STATE\_STAGE);// 切換至GAME\_STATE\_STAGE

}

else if (x == 285) {

ofstream ofs;

ofs.open("store2.txt");

ofs.ios\_base::trunc;

ofs << MaxLevel << endl;

ofs.close();

GotoGameState(GAME\_STATE\_STAGE);// 切換至GAME\_STATE\_STAGE

}

else {

ofstream ofs;

ofs.open("store3.txt");

ofs.ios\_base::trunc;

ofs << MaxLevel << endl;

ofs.close();

GotoGameState(GAME\_STATE\_STAGE);//切換至GAME\_STATE\_STAGE

}

}

}

else if (nChar == KEY\_LEFT) {

if (x > 85)

x -= 200;

select.SetTopLeft(x, y);

}

else if (nChar == KEY\_RIGHT) {

if (x < 485)

x += 200;

select.SetTopLeft(x, y);

}

else if (nChar == KEY\_ESC)

GotoGameState(GAME\_STATE\_INIT);// 切換至GAME\_STATE\_INIT

}

void CGameStateStore::OnLButtonDown(UINT nFlags, CPoint point)

{

}

void CGameStateStore::OnMove() {

}

void CGameStateStore::OnShow()

{

CDC\* pDC = CDDraw::GetBackCDC();

CFont f, \* fpq;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fpq = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 200));

pDC->TextOut(0, 0, "ESC返回開始畫面");

pDC->SelectObject(fpq); // 放掉 font f

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

bg.ShowBitmap();

select.ShowBitmap();

store1.ShowBitmap();

store2.ShowBitmap();

store3.ShowBitmap();

}

/////////////////////////////////////////////////////////////////////////////

// 這個class為遊戲的結束狀態(Game Over)

/////////////////////////////////////////////////////////////////////////////

CGameStateOver::CGameStateOver(CGame\* g)

: CGameState(g)

{

}

void CGameStateOver::OnMove()

{

counter--;

if (counter < 0)

GotoGameState(GAME\_STATE\_STAGE);

}

void CGameStateOver::OnBeginState()

{

counter = 30 \* 5; // 5 seconds

}

void CGameStateOver::OnInit()

{

//

// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人

// 等的不耐煩，遊戲會出現「Loading ...」，顯示Loading的進度。

//

ShowInitProgress(66); // 接個前一個狀態的進度，此處進度視為66%

//

// 開始載入資料

//

Sleep(300); // 放慢，以便看清楚進度，實際遊戲請刪除此Sleep

//

// 最終進度為100%

//

ShowInitProgress(100);

}

void CGameStateOver::OnShow()

{

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 0));

char str[80]; // Demo 數字對字串的轉換

sprintf(str, "廢物! (%d)", counter / 30);

pDC->TextOut(240, 210, str);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

CGameStatePass::CGameStatePass(CGame\* g)

: CGameState(g)

{

}

void CGameStatePass::OnMove()

{

counter--;

if (counter < 0)

GotoGameState(GAME\_STATE\_STAGE);

}

void CGameStatePass::OnBeginState()

{

counter = 30 \* 5; // 5 seconds

if (MaxLevel <= level) {

MaxLevel = level;

}

}

void CGameStatePass::OnInit()

{

ShowInitProgress(66);

Sleep(300);

ShowInitProgress(100);

}

void CGameStatePass::OnShow()

{

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 0));

char str[80];

//藉由分數顯示不同評級

if(getscore == 100)

sprintf(str, "你的評級為S ");

else if(getscore >90)

sprintf(str, "你的評級為A ");

else if(getscore >70)

sprintf(str, "你的評級為B ");

else if(getscore >50)

sprintf(str, "你的評級為C ");

else

sprintf(str, "你好爛 " );

pDC->TextOut(240, 210, str);

pDC->SelectObject(fp); // 放掉 font f

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

/////////////////////////////////////////////////////////////////////////////

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

/////////////////////////////////////////////////////////////////////////////

CGameStateRun::CGameStateRun(CGame\* g)

: CGameState(g)

{

}

CGameStateRun::~CGameStateRun()

{

}

void CGameStateRun::OnBeginState()

{

LoadKeyboardLayout("0x0409", KLF\_ACTIVATE | KLF\_SETFORPROCESS);

const int BALL\_GAP = 90;

const int BALL\_XY\_OFFSET = 45;

const int BALL\_PER\_ROW = 7;

const int HITS\_LEFT = 5; //初始分數

int CLOCK = start; //時間

const int HITS\_LEFT\_X = 590;

const int HITS\_LEFT\_Y = 0;

const int BACKGROUND\_X = 60;

const int ANIMATION\_SPEED = 15;

//一開始清除所有上一次關卡的資料

beat\_x = 0;

clap.clear();

first.clear();

everytime.clear();

location.clear();

jump\_time\_list.clear();

noise.SetXY(0, 0);

noise.SetIsAlive(true);

noise.SetIsShow(false);

test1.SetXY(425, 0);

test1.SetIsAlive(true);

test1.SetIsShow(false);

bg1.SetDelayCount(3);

hand.SetDelayCount(1);

background.SetTopLeft(BACKGROUND\_X, 0); // 設定背景的起始座標

clocktime.SetInteger(CLOCK);

clocktime.SetTopLeft(0, 400);

hits\_left.SetInteger(HITS\_LEFT); // 指定剩下的撞擊數

hits\_left.SetTopLeft(HITS\_LEFT\_X, HITS\_LEFT\_Y); // 指定剩下撞擊數的座標

CAudio::Instance()->Play(AUDIO\_DING, false); // 撥放 WAVE

//判斷屬於什麼關卡 載入不同資料

//一開始先設定location 每次音符出現的位置

//jump\_time\_list 每次音符的跳動次數

//everytime 分割出的時間陣列

if (level == 1) {

CAudio::Instance()->Play(AUDIO\_ONE, false); // 撥放 MIDI

stage1 data;

for (int i = 0; i < data.total\_tap; i++) {

clap.push\_back(data.clap[i]-250);

first.push\_back(data.first[i]-250);

}

total\_tap = data.total\_tap;

for (int i = 0; i < total\_tap; i++) {

for (int j = 1; j <= 7; j++) {

location.push\_back(j);

jump\_time\_list.push\_back(1);

}

}

for (int i = 0; i < total\_tap; i++)

{

int interval = (clap[i] - first[i]) / 6;

int s = first[i];

for (int j = 0; j < 6; j++)

{

everytime.push\_back(s);

s += interval;

}

everytime.push\_back(clap[i]);

}

everytime.push\_back(999999999);

}

else if (level == 2) {

CAudio::Instance()->Play(AUDIO\_TWO, false); // 撥放 MIDI

stage2 data;

for (int i = 0; i < data.total\_tap; i++) {

clap.push\_back(data.clap[i]-100);

first.push\_back(data.first[i]-100);

}

total\_tap = data.total\_tap;

for (int i = 0; i < total\_tap; i++) {

for (int j = 1; j <= 7; j++) {

location.push\_back(j);

jump\_time\_list.push\_back(1);

}

}

for (int i = 0; i < total\_tap; i++)

{

int interval = (clap[i] - first[i]) / 6;

int s = first[i];

for (int j = 0; j < 6; j++)

{

everytime.push\_back(s);

s += interval;

}

everytime.push\_back(clap[i]);

}

everytime.push\_back(999999999);

}

else if (level == 3) {

//level 3特別複雜 很多特殊狀況 需要特別設定

CAudio::Instance()->Play(AUDIO\_THREE, false); // 撥放 MIDI

stage3 data;

for (int i = 0; i < data.total\_tap; i++) {

clap.push\_back(data.clap[i]-100);

first.push\_back(data.first[i]-100);

}

total\_tap = data.total\_tap;

for (int i = 0; i < total\_tap; i++) {

if (i == 3) {

for (int j = 1; j <= 7; j++) {

location.push\_back(j);

if (j == 3)

jump\_time\_list.push\_back(12);

else if(j == 7)

jump\_time\_list.push\_back(4);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 4) {

for (int j = 1; j <= 7; j++) {

location.push\_back(j);

if (j == 6)

jump\_time\_list.push\_back(12);

else if (j == 7)

jump\_time\_list.push\_back(16);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 6) {

int lo[] = { 1,2,3,4,5,4,5,7 };

for (int j = 0; j <8; j++) {

location.push\_back(lo[j]);

jump\_time\_list.push\_back(1);

}

}

else if (i == 7) {

int lo[] = { 1,2,3,5,7 };

for (int j = 0; j <5; j++) {

location.push\_back(lo[j]);

if(3== lo[j])

jump\_time\_list.push\_back(8);

else if(5 == lo[j])

jump\_time\_list.push\_back(8);

else if(7 == lo[j])

jump\_time\_list.push\_back(16);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 11) {

for (int j = 1; j <= 7; j++) {

location.push\_back(j);

if (j == 3)

jump\_time\_list.push\_back(12);

else if (j == 6)

jump\_time\_list.push\_back(8);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 12) {

int lo[] = { 1,2,3,2,4,5,6,7 };

for (int j = 0; j <8; j++) {

location.push\_back(lo[j]);

if (j == 6)

jump\_time\_list.push\_back(8);

else if (j == 7)

jump\_time\_list.push\_back(8);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 13) {

int lo[] = { 1,3,1,2,4,2,5,6,7 };

for (int j = 0; j < 9; j++) {

location.push\_back(lo[j]);

if (j == 8)

jump\_time\_list.push\_back(8);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 14) {

int lo[] = { 1,3,1,2,3,2,4,6,7 };

for (int j = 0; j < 9; j++) {

location.push\_back(lo[j]);

jump\_time\_list.push\_back(1);

}

}

else if (i == 15) {

int lo[] = { 1,2,3,2,1,4,5,6,7 };

for (int j = 0; j < 9; j++) {

location.push\_back(lo[j]);

jump\_time\_list.push\_back(1);

}

}

else if (i == 19) {

int lo[] = { 1,2,3,4,5,3,6,7 };

for (int j = 0; j < 8; j++) {

location.push\_back(lo[j]);

if( j == 2 || j == 5)

jump\_time\_list.push\_back(8);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 23) {

int lo[] = { 1,5,7 };

for (int j = 0; j < 3; j++) {

location.push\_back(lo[j]);

if (j ==0)

jump\_time\_list.push\_back(24);

else if ( j ==1)

jump\_time\_list.push\_back(8);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 26) {

int lo[] = { 1,2,3,1,4,5,6,7 };

for (int j = 0; j < 8; j++) {

location.push\_back(lo[j]);

if (j == 1)

jump\_time\_list.push\_back(8);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 27 || i == 28) {

int lo[] = { 1,2,3,1,4,1,6,7 };

for (int j = 0; j < 8; j++) {

location.push\_back(lo[j]);

if (j == 7)

jump\_time\_list.push\_back(8);

else

jump\_time\_list.push\_back(1);

}

}

else if (i == 31) {

int lo[] = { 1,2,3,1,4,1,6,7 };

for (int j = 0; j < 8; j++) {

location.push\_back(lo[j]);

jump\_time\_list.push\_back(1);

}

}

else if (i == 32) {

int lo[] = { 1,2,1,1,4,5,6,7 };

for (int j = 0; j < 8; j++) {

location.push\_back(lo[j]);

jump\_time\_list.push\_back(1);

}

}

else if (i == 33) {

int lo[] = { 1,5,4,5,6,7 };

for (int j = 0; j < 6; j++) {

if (j == 0) {

jump\_time\_list.push\_back(24);

}

else {

jump\_time\_list.push\_back(1);

}

location.push\_back(lo[j]);

}

}

else if (i == 34) {

int lo[] = { 1,1,2,3,4,1,5,6,7 };

for (int j = 0; j < 9; j++) {

location.push\_back(lo[j]);

jump\_time\_list.push\_back(1);

}

}

else {

for (int j = 1; j <= 7; j++) {

if (i == 5 && j == 6) {

location.push\_back(5);

jump\_time\_list.push\_back(1);

}

else if (i == 9 && j == 7) {

location.push\_back(j);

jump\_time\_list.push\_back(12);

}

else if (i == 10 && j == 7) {

location.push\_back(j);

jump\_time\_list.push\_back(8);

}

else if (i == 17 && j == 7) {

location.push\_back(j);

jump\_time\_list.push\_back(20);

}

else if (i == 24 && j == 7) {

location.push\_back(j);

jump\_time\_list.push\_back(8);

}

else if (i == 25 && j == 3) {

location.push\_back(j);

jump\_time\_list.push\_back(12);

}

else if (i == 30 && j == 5) {

location.push\_back(1);

jump\_time\_list.push\_back(1);

}

else {

location.push\_back(j);

jump\_time\_list.push\_back(1);

}

}

}

}

for (int i = 0; i < total\_tap; i++)

{

int len = 6;

if (i == 6 || i ==12 || i == 19 || i ==26 || i == 27 || i == 28 || i ==31 || i ==32) {

len = 7;

}

else if (i == 13 || i == 14 || i == 15 || i == 34) {

len = 8;

}

else if (i == 23) {

len = 2;

}

else if (i == 33) {

len = 5;

}

else if (i == 7) {

len = 4;

}

int interval = (clap[i] - first[i]) / len;

int s = first[i];

for (int j = 0; j < len; j++)

{

everytime.push\_back(s);

s += interval;

}

everytime.push\_back(clap[i]);

}

everytime.push\_back(999999999);

}

else if (level == 4) {

CAudio::Instance()->Play(AUDIO\_FOUR, false); // 撥放 MIDI

stage4 data;

for (int i = 0; i < data.total\_tap; i++) {

clap.push\_back(data.clap[i]);

first.push\_back(data.first[i]);

}

total\_tap = data.total\_tap;

for (int i = 0; i < total\_tap; i++) {

for (int j = 1; j <= 30; j++) {

if (j < 21)

location.push\_back(1);

else if (j < 26)

location.push\_back(2);

else

location.push\_back(j-25+2);

if (j == 1)

jump\_time\_list.push\_back(2);

else

jump\_time\_list.push\_back(1);

}

}

for (int i = 0; i < total\_tap; i++)

{

int interval = (clap[i] - first[i]) / 29;

int s = first[i];

for (int j = 0; j < 29; j++)

{

everytime.push\_back(s);

s += interval;

}

everytime.push\_back(clap[i]);

}

everytime.push\_back(999999999);

}

else if (level == 5) {

CAudio::Instance()->Play(AUDIO\_FIVE, false); // 撥放 MIDI

stage5 data;

for (int i = 0; i < data.total\_tap; i++) {

clap.push\_back(data.clap[i]-100);

first.push\_back(data.first[i]-100);

}

total\_tap = data.total\_tap;

for (int i = 0; i < total\_tap; i++) {

if (i < 21) {

for (int j = 1; j <= 30; j++) {

if (j < 21)

location.push\_back(1);

else if (j < 26)

location.push\_back(2);

else

location.push\_back(j - 25 + 2);

if (j == 1)

jump\_time\_list.push\_back(2);

else

jump\_time\_list.push\_back(1);

}

}

else{

for (int j = 1; j <= 10; j++) {

if (j < 5)

location.push\_back(1);

else if (j < 6)

location.push\_back(2);

else

location.push\_back(j - 5 + 2);

if (j == 1)

jump\_time\_list.push\_back(2);

else

jump\_time\_list.push\_back(1);

}

}

}

for (int i = 0; i < total\_tap; i++)

{

if (i < 21) {

int interval = (clap[i] - first[i]) / 29;

int s = first[i];

for (int j = 0; j < 29; j++)

{

everytime.push\_back(s);

s += interval;

}

everytime.push\_back(clap[i]);

}

else {

int interval = (clap[i] - first[i]) / 9;

int s = first[i];

for (int j = 0; j < 9; j++)

{

everytime.push\_back(s);

s += interval;

}

everytime.push\_back(clap[i]);

}

}

everytime.push\_back(999999999);

}

else if (level == 6) {

CAudio::Instance()->Play(AUDIO\_SIX, false); // 撥放 MIDI

stage6 data;

for (int i = 0; i < data.total\_tap; i++) {

clap.push\_back(data.clap[i]-100);

first.push\_back(data.first[i]-100);

}

total\_tap = data.total\_tap;

for (int i = 0; i < total\_tap; i++) {

for (int j = 1; j <= 20; j++) {

if (j < 11)

location.push\_back(1);

else if (j < 16)

location.push\_back(2);

else

location.push\_back(j - 15 + 2);

if (j == 1)

jump\_time\_list.push\_back(2);

else

jump\_time\_list.push\_back(1);

}

}

for (int i = 0; i < total\_tap; i++)

{

int interval = (clap[i] - first[i]) / 19;

int s = first[i];

for (int j = 0; j < 19; j++)

{

everytime.push\_back(s);

s += interval;

}

everytime.push\_back(clap[i]);

}

everytime.push\_back(999999999);

}

}

ofstream ofs;

int tt = 210;

void CGameStateRun::OnMove() // 移動遊戲元素

{

//各關卡內一些動畫的行為

if(level==1)background1.OnMove();

if (level == 3 && c>=30 && c<31 && start<clap[31] ) {

noise.SetIsShow(true);

}

if (level == 3) {

bg1.OnMove();

if (tt == 210 || start < clap[c])

tt = 220;

else

tt = 210;

}

if (level == 5) {

bg3.OnMove();

if (tt == 210 || start<clap[c])

tt = 220;

else

tt = 210;

}

if (level == 6) {

if (tt == 210 || start < clap[c])

tt = 220;

else

tt = 210;

if (c > 74 && c < 109) {

c\_practice.OnJump2();

}

}

if (level == 6)bg4.OnMove();

start = (clock()-END)-600;

int const min = 20;

int const max = 480;

int const minx = 0;

int const maxx = 300;

//如果時間到一個時間點 音符就會換位置

if (start > everytime[beat\_x]-100) {

c\_practice.OnMove(location[beat\_x]);

jump\_time = jump\_time\_list[beat\_x]\*2;

beat\_x++;

}

if (jump\_time) {

//jump方式

c\_practice.OnJump();

jump\_time--;

}

}

void CGameStateRun::OnInit() // 遊戲的初值及圖形設定

{

ShowInitProgress(33); // 接個前一個狀態的進度，此處進度視為33%

//load遊戲共同動畫還有各關動畫 圖片

noise.LoadBitmapA(IDB\_BITMAP61);

isClick = false;

test1.LoadBitmap(177);

bg.LoadBitmap(IDB\_BITMAP49);

for (int i = IDB\_BITMAP62; i <= IDB\_BITMAP71; i++) {

bg1.AddBitmap(i);

}

bg2.LoadBitmap(IDB\_BITMAP52);

bg3.AddBitmap(IDB\_BITMAP52);

bg3.AddBitmap(IDB\_BITMAP53);

for (int i = IDB\_BITMAP54; i <= IDB\_BITMAP59; i++) {

bg4.AddBitmap(i);

}

bg3.SetDelayCount(2);

bg4.SetDelayCount(2);

background.LoadBitmap(IDB\_BACKGROUND); // 載入背景的圖形

background1.AddBitmap(IDB\_BG1);

background1.AddBitmap(IDB\_BG2);

background1.SetDelayCount(7);

ShowInitProgress(50);

bar.LoadBitmap(IDB\_BITMAP17,RGB(255, 255, 255));

hand.AddBitmap(IDB\_HAND1,RGB(255, 255, 255));

hand.AddBitmap(IDB\_HAND2, RGB(255, 255, 255));

test.LoadBitmap(IDB\_INITSELECTBOX);

xx1.LoadBitmap(IDB\_BITMAP30,RGB(0,0,0));

xx2.LoadBitmap(IDB\_BITMAP30, RGB(0, 0, 0));

help.LoadBitmap(IDB\_HELP, RGB(255, 255, 255)); // 載入說明的圖形

corner.LoadBitmap(IDB\_CORNER); // 載入角落圖形 // 載入圖形

hits\_left.LoadBitmap();

c\_practice.LoadBitmap();

//載入各關音樂

CAudio::Instance()->Load(AUDIO\_ONE, "sounds\\1.mp3");

CAudio::Instance()->Load(AUDIO\_TWO, "sounds\\2.mp3");

CAudio::Instance()->Load(AUDIO\_THREE, "sounds\\3.mp3");

CAudio::Instance()->Load(AUDIO\_FOUR, "sounds\\4.mp3");

CAudio::Instance()->Load(AUDIO\_FIVE, "sounds\\5.mp3");

CAudio::Instance()->Load(AUDIO\_SIX, "sounds\\6.mp3");

}

void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

const char KEY\_SPACE = ' ';

if (nChar == KEY\_SPACE) {

hand.OnMove();

}

}

bool H = false;

void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

const char KEY\_SPACE = ' '; //空白鍵

const char skip = 83; //SKIP 按s使用跳關密技

if (nChar == KEY\_SPACE) {

//按空白鍵後判斷時間有沒有在評分範圍

isClick = true;

if (start - 300 < clap[c] && clap[c] < start + 300) {

isGet = true;

}

else {

isGet = false;

}

test1.SetIsShow(true);

hand.OnMove();

hand.Reset();

int x = rand() % (400 - 20 + 1) + 20;

int y = rand() % (480 + 1) + 0;

background.SetTopLeft(x, y);

}

else if (nChar == skip) {

getscore = 100;

if (level == 1) {

CAudio::Instance()->Stop(AUDIO\_ONE); // 停止 MIDI

}

else if (level == 2) {

CAudio::Instance()->Stop(AUDIO\_TWO); // 停止 MIDI

}

else if (level == 3) {

CAudio::Instance()->Stop(AUDIO\_THREE); // 停止 MIDI

}

else if (level == 4) {

CAudio::Instance()->Stop(AUDIO\_FOUR); // 停止 MIDI

}

else if (level == 5) {

CAudio::Instance()->Stop(AUDIO\_FIVE); // 停止 MIDI

}

else if (level == 6) {

CAudio::Instance()->Stop(AUDIO\_SIX); // 停止 MIDI

}

c = 0;

GotoGameState(GAME\_STATE\_PASS);

}

}

void CGameStateRun::OnLButtonDown(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

}

void CGameStateRun::OnLButtonUp(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

}

void CGameStateRun::OnMouseMove(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

}

void CGameStateRun::OnRButtonDown(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

}

void CGameStateRun::OnRButtonUp(UINT nFlags, CPoint point) // 處理滑鼠的動作

{

}

void CGameStateRun::OnShow()

{

if (level == 1) {

background1.OnShow();

}

else if(level ==2) {

bg.ShowBitmap();

}

else if(level == 3){

bg1.OnShow();

}

else if (level == 4) {

bg2.ShowBitmap();

}

else if (level == 5) {

bg3.OnShow();

}

else if (level == 6) {

bg4.OnShow();

}

if (level != 5)bar.SetTopLeft(0, 210);

else if(level ==5 || level==6) bar.SetTopLeft(0, tt);

hits\_left.ShowBitmap();

bar.ShowBitmap();

if (test1.IsShow()) {

test1.OnShow();

}

if (clap[c]+100 <= start) { //到了一個時間點才會做判斷 而不是按空白鍵做判斷

if (!isClick) {

hits\_left.Add(-1); //如果沒按空白鍵就會扣分

}

else {

if (isGet) {

hits\_left.Add(1); //如果按在正確的時間則會加分

}

else

{

hits\_left.Add(-1); //否則扣分

}

}

c++;

isClick = false;

if (hits\_left.GetInteger() <= 0) { //分數被扣光

if (level == 1) {

CAudio::Instance()->Stop(AUDIO\_ONE); // 停止 MIDI

}

else if (level == 2) {

CAudio::Instance()->Stop(AUDIO\_TWO); // 停止 MIDI

}

else if (level == 3) {

CAudio::Instance()->Stop(AUDIO\_THREE); // 停止 MIDI

}

else if (level == 4) {

CAudio::Instance()->Stop(AUDIO\_FOUR); // 停止 MIDI

}

else if (level == 5) {

CAudio::Instance()->Stop(AUDIO\_FIVE); // 停止 MIDI

}

else if (level == 6) {

CAudio::Instance()->Stop(AUDIO\_SIX); // 停止 MIDI

}

c = 0;

GotoGameState(GAME\_STATE\_OVER);

}

if (c == total\_tap) { //時間跑到遊戲結束

if (level == 1) {

CAudio::Instance()->Stop(AUDIO\_ONE); // 停止 MIDI

}

else if (level == 2) {

CAudio::Instance()->Stop(AUDIO\_TWO); // 停止 MIDI

}

else if (level == 3) {

CAudio::Instance()->Stop(AUDIO\_THREE); // 停止 MIDI

}

else if (level == 4) {

CAudio::Instance()->Stop(AUDIO\_FOUR); // 停止 MIDI

}

else if (level == 5) {

CAudio::Instance()->Stop(AUDIO\_FIVE); // 停止 MIDI

}

else if (level == 6) {

CAudio::Instance()->Stop(AUDIO\_SIX); // 停止 MIDI

}

getscore = (hits\_left.GetInteger() \* 100 / (total\_tap+5)); //結算獲得的分數

c = 0;

GotoGameState(GAME\_STATE\_PASS);//到gamestatepass

}

}

test1.SetIsShow(false);

if (start > everytime[0]) //到第一個時間點才顯示音符

c\_practice.OnShow();

if (start > everytime[28] - 100 && level == 2) { //level2的特別畫面

xx1.SetTopLeft(310, 220);

xx1.ShowBitmap();

xx2.SetTopLeft(370, 220);

xx2.ShowBitmap();

}

hand.SetTopLeft(640 - 479, 480 - 75);

hand.OnShow();

if (noise.IsShow()) {

noise.OnShow();

}

noise.SetIsShow(false);

corner.SetTopLeft(0, 0);

corner.SetTopLeft(SIZE\_X - corner.Width(), SIZE\_Y - corner.Height());

}

}

mygame.h

/\*

\* mygame.h: 本檔案儲遊戲本身的class的interface

\* Copyright (C) 2002-2008 Woei-Kae Chen <wkc@csie.ntut.edu.tw>

\*

\* This file is part of game, a free game development framework for windows.

\*

\* game is free software; you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation; either version 2 of the License, or

\* (at your option) any later version.

\*

\* game is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

\*

\* 2004-03-02 V4.0

\* 1. Add CGameStateInit, CGameStateRun, and CGameStateOver to

\* demonstrate the use of states.

\* 2005-09-13

\* Rewrite the codes for CBall and CEraser.

\* 2005-09-20 V4.2Beta1.

\* 2005-09-29 V4.2Beta2.

\* 2006-02-08 V4.2

\* 1. Rename OnInitialUpdate() -> OnInit().

\* 2. Replace AUDIO\_CANYON as AUDIO\_NTUT.

\* 3. Add help bitmap to CGameStateRun.

\* 2006-09-09 V4.3

\* 1. Rename Move() and Show() as OnMove and OnShow() to emphasize that they are

\* event driven.

\* 2008-02-15 V4.4

\* 1. Add namespace game\_framework.

\* 2. Replace the demonstration of animation as a new bouncing ball.

\* 3. Use ShowInitProgress(percent) to display loading progress.

\*/

#include "CEraser.h"

#include "CBall.h"

#include "CBouncingBall.h"

#include "ClongGray.h"

#include "stage1.h"

#include "stage2.h"

#include "stage3.h"

#include "stage4.h"

#include "stage5.h"

#include "stage6.h"

namespace game\_framework {

enum AUDIO\_ID { // 定義各種音效的編號

AUDIO\_DING, // 0

AUDIO\_LAKE, // 1

AUDIO\_NTUT, // 2

AUDIO\_CLICK,

AUDIO\_TEST,

AUDIO\_ONE, //關卡音樂1~6

AUDIO\_TWO,

AUDIO\_THREE,

AUDIO\_FOUR,

AUDIO\_FIVE,

AUDIO\_SIX

};

class CBouncingBall;

class CPractice {

public:

CPractice();

void LoadBitmap();

void OnMove(int );

void OnShow();

void OnJump();

void OnJump2();

int getX();

private:

CMovingBitmap pic;

int x, y;

};

/////////////////////////////////////////////////////////////////////////////

// 這個class為遊戲的遊戲開頭畫面物件

// 每個Member function的Implementation都要弄懂

/////////////////////////////////////////////////////////////////////////////

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤Up的動作

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

void OnMove();

private:

CAnimation logo; // csie的logo

CMovingBitmap title;

CMovingBitmap BG;

ClongGray about;

CAnimation title1;

int x,y;

};

class CGameStateStage : public CGameState {

public:

CGameStateStage(CGame\* g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重新載入所需的變數

void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤Up的動作

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

void OnMove();

private:

CMovingBitmap bg; //背景圖片

CAnimation stage11; //關卡1~3的動畫

CAnimation stage12;

CAnimation stage13;

CMovingBitmap select;

CPractice test11; //選擇關卡的圖示

int x, y; //選擇關卡的座標

};

class CGameStateStage2 : public CGameState {

public:

public:

CGameStateStage2(CGame\* g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重新載入所需的變數

void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤Up的動作

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

void OnMove();

private:

CMovingBitmap bg; //背景圖片

CMovingBitmap stage11; //關卡4~6的圖片

CMovingBitmap stage12;

CMovingBitmap stage13;

CMovingBitmap select;

CPractice test11; //選擇關卡的圖示

int x, y; //選擇關卡的座標

};

class CGameStateStore : public CGameState {

public:

CGameStateStore(CGame\* g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次載入所需的變數

void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤Up的動作

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

void OnMove();

private:

CMovingBitmap bg; //背景圖片

CMovingBitmap store1; //關卡1~3的動畫

CMovingBitmap store2;

CMovingBitmap store3;

CMovingBitmap select;

CPractice test11; //選擇關卡的圖示

int x, y; //選擇關卡的座標

};

/////////////////////////////////////////////////////////////////////////////

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

// 每個Member function的Implementation都要弄懂

/////////////////////////////////////////////////////////////////////////////

class CGameStateRun : public CGameState {

public:

CGameStateRun(CGame \*g);

~CGameStateRun();

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit(); // 遊戲的初值及圖形設定

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnLButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CMovingBitmap background; // 背景圖

CMovingBitmap bg; //關卡背景1~6

CAnimation bg1;

CMovingBitmap bg2;

CAnimation bg3;

CAnimation bg4;

CMovingBitmap bg5;

CMovingBitmap help; // 說明圖

CMovingBitmap test;

CMovingBitmap corner; // 角落圖

CMovingBitmap hand1;

CMovingBitmap xx1;

CMovingBitmap xx2;

ClongGray noise;

ClongGray hand2;

CInteger hits\_left; // 分數

CInteger num;

CInteger clocktime;

CMovingBitmap long\_gray; //案空白鍵會出現的長條

CPractice c\_practice; //音符

ClongGray test1;

vector<int> everytime; //時間列表

vector<int> location; //位置列表

vector<int> jump\_time\_list; //跳動次數列表

bool isClick; //是否在判斷分數範圍及有沒有獲得分數

bool isGet;

CAnimation hand; //關卡內部動畫

CMovingBitmap bar;

CAnimation background1;

CAnimation tempo;

int beat\_x = 1; //時間陣列的index

int c = 0; //遊戲總時間位置

vector<int> clap; //第7拍時間

vector<int> first; //第1拍時間

int total\_tap; //總分

int jump\_time=2; //預設跳動數

};

/////////////////////////////////////////////////////////////////////////////

// 這個class為遊戲的結束狀態(Game Over)

// 每個Member function的Implementation都要弄懂

/////////////////////////////////////////////////////////////////////////////

class CGameStateOver : public CGameState {

public:

CGameStateOver(CGame \*g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int counter; // 倒數之計數器

};

class CGameStatePass : public CGameState {

public:

CGameStatePass(CGame\* g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int counter; // 倒數之計數器

};

}

stage1.h

namespace game\_framework {

//儲存關卡1資訊

class stage1

{

public:

int total\_tap = 44;

int clap[44] = { 11008,13416,15823,17111,18266,20691,23069,25519,26706,27896,29349,30570,31792,32951,34207,35364,36555,37780,38968,40120,41313,42538,43761,44917,45969,47365,48622,49776,51032,52219,53411,54535,55728,56917,58110,59506,60587,61908,62962,64253,66323,67518,68676,72779 };

int first[44] = { 9512, 11955, 14295, 16454, 17744, 19168, 21541, 23916, 26083, 27299, 28757, 30077, 31194, 32413, 33601, 34818, 35999, 37258, 38408, 39592, 40812, 41962, 43251, 44432, 45619, 46809, 47997, 49213, 50397, 51623, 52816, 54031, 55215, 56434, 57620, 58942, 60131, 61419, 62572, 63961, 66026, 67144, 68366, 71924 };

protected:

};

}

stage2.h

namespace game\_framework {

//儲存關卡2資訊

class stage2

{

public:

int total\_tap = 23;

int clap[44] = { 14127,16740,19441,22086,24788,28755,31436,34077,36819,39453,42059,44599,46464,50021,52794,55440,57100,60791,63428,66096,67751,71441,73166 };

int first[44] = { 12105,14778,17448,20092,22798,26932,29403,32084,34793,37438,40051,42798,45028,48310,50775,53413,55342,58769,61406,64088,66015,69595,71864 };

protected:

};

}

stage3.h

namespace game\_framework {

//儲存關卡3資訊

class stage3

{

public:

int total\_tap = 44;

int clap[44] = { 18006,20749,23395,26112,28617,31358,33961,36702,39303,41971,44677,47350,50015,52720,55433,58036,71374,74046,76764,79431,103425,106000,108745,111417,113989,116731,119334,122041,124578,127388,129998,132639,135316,138155,140763,143437 };

int first[44] = { 15928,18735,21381,24058,26630,29412,32016,34688,37329,40034,42640,45381,48022,50665,53336,56071,69560,72036,74677,77355,101290,104096,106698,109375,112015,114695,117406,120015,122721,125331,127294,130645,133313,136021,138725,141364 };

protected:

};

}

stage4.h

namespace game\_framework {

//儲存關卡4資訊

class stage4

{

public:

int total\_tap = 59;

int clap[59] = { 8737,10291,11915,13471,15093,16649,18308,19873,21466,23089,24715,26305,27963,29519,31111,32664,34323,35881,37509,39069,40727,42284,43905,45530,47118,48740,50365,51923,53519,55077,56698,58285,71175,72739,74337,75892,77518,79147,80740,82329,83921,85474,87038,88660,90319,91949,93473,95092,96676,98367,99929,101523,103177,104764,106354,107909,109496,111224,112781 };

int first[59] = { 8012,9504,11093,12483,14275,15903,17467,18684,20610,22140,23867,25224,27080,28681,30234,31553,33483,35008,36634,38231,39859,41381,43040,44354,46322,47884,49474,50895,52622,53979,55709,57169,70430,71956,73380,75044,76395,78250,79570,81497,82823,84623,85938,87871,89123,90879,92332,94095,95582,97443,98834,100530,101954,103748,105138,107069,108050,110253,111907 };

protected:

};

}

stage5.h

namespace game\_framework {

//儲存關卡5資訊

class stage5

{

public:

int total\_tap = 114;

int first[114] = { 12420,15098,17798,20510,23257,24645,26001,27365,28658,30079,31506,32802,34154,35545,36835,38225,39547,40973,42299,43721,45073,45818,46496,47177,47853,48528,49201,49882,50596,51274,51922,52634,53313,53985,54663,55337,56049,56726,57438,58083,58760,59476,60155,60800,61512,62156,62831,63513,64191,64906,65588,66266,66943,68330,69652,71040,72395,73755,75080,76500,77795,78541,79218,79930,80605,81283,81962,82608,83320,83962,84673,85355,86033,86708,87416,88129,88671,89150,89487,89825,90128,90469,90773,91149,91521,91827,92166,92507,92814,93117,93422,93728,94066,94442,94781,95120,95458,95797,96135,96473,96777,97116,97453,97760,98102,98408,98813,99150,100000,102404,105150,106220,107190,109000 };

int clap[114] = { 13710,16358,19103,21779,23822,25241,26631,27954,29348,30742,32094,33412,34832,36182,37542,38870,40255,41582,43013,44308,45430,46106,46782,47459,48135,48811,49488,50134,50844,51489,52165,52876,53592,54263,54941,55615,56293,57008,57683,58366,59009,59690,60368,61078,61722,62402,63045,63793,64472,65152,65829,66506,67689,68872,70226,71580,72974,74294,75650,77072,78116,78725,79444,80125,80805,81522,82237,82918,83562,84238,84952,85667,86308,87022,87697,88307,88782,89157,89565,89939,90279,90616,90955,91262,91603,91941,92281,92624,92927,93220,93540,94848,94206,94580,94900,95240,95578,95917,96255,96593,96897,97236,97573,97883,98222,98553,98923,99258,101174,103665,106406,107722,109022,110000 };

protected:

};

}

stage6.h

namespace game\_framework {

//儲存關卡4資訊

class stage6

{

public:

int total\_tap = 118;

int first[118] = { 14297,15922,17540,19234,20834,22463,24091,25786,27408,29067,30696,32457,34083,35637,37295,38992,40678,42342,43967,45594,47285,48911,50538,52266,53857,55556,57176,58841,60498,62234,63928,65521,67208,68870,70507,72204,73801,75425,77087,78705,80395,82089,83725,85379,87004,88734,90352,91879,95337,97031,98588,100318,101946,103536,105197,106889,108511,110136,111829,113522,115113,116801,118594,120150,121807,123470,125131,126787,128440,130139,131770,133358,135087,136680,138342,140001,141659,143318,144945,146585,161537,162419,163234,164011,164824,165674,166522,167369,168244,169057,169869,170714,171526,171966,172407,172814,173219,173590,173996,174369,175505,176435,178095,179720,181384,183007,184734,186361,187989,189617,191273,192940,194603,196324,197917,199476,201203,202867 };

int clap[118] = { 15079,16670,18301,19992,21586,23214,24938,26532,28227,29892,31522,33183,34846,36439,38135,39794,41459,43152,44778,46410,48040,49799,51359,53058,54717,56374,57970,59664,61256,62984,64681,66307,67932,69629,71252,72947,74613,76276,77939,79607,81206,82935,84596,86249,87914,89541,91169,92774,96135,97798,99392,101053,102714,104379,106047,107713,109367,110995,112614,114311,115871,117672,119298,120990,122615,124275,125871,127538,129234,130825,132488,134187,135889,137618,139177,140842,142505,144098,145827,147384,161987,162835,163651,164467,165276,166053,166832,167681,168595,169406,170288,170999,171747,172219,172692,173100,173506,173948,174354,175004,176012,177164,178894,180488,182189,183882,185544,187171,188765,190424,192116,193775,195469,197030,198723,200414,201974,203666};

protected:

};

}

ClongGray.h

namespace game\_framework {

//節奏醫生中按空白鍵會跑出來的物件

class ClongGray

{

public:

ClongGray();

bool IsAlive(); // 是否活著

bool IsShow();

void LoadBitmap(int x); // 載入圖形

//void OnMove(); // 移動

void OnShow(); // 將圖形貼到畫面

void SetXY(int nx, int ny); // 設定座標

void SetIsAlive(bool alive); // 設定是否活著

void SetIsShow(bool show);

//void SetDelay(int d);

protected:

CMovingBitmap bmp; // 圖

//CMovingBitmap bmp\_center; // 圓心的圖

int x, y; // 圖的座標

bool is\_alive; // 是否活著

bool is\_show;

};

}

CloneGray.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CEraser.h"

#include "CBall.h"

#include "ClongGray.h"

namespace game\_framework {

ClongGray::ClongGray() {

is\_alive = true;

x = y = 0;

}

bool ClongGray::IsAlive() {

return is\_alive;

}

bool ClongGray::IsShow() {

return is\_show;

}

void ClongGray::LoadBitmap(int x) {

bmp.LoadBitmap(x);

}

void ClongGray::SetIsAlive(bool alive)

{

is\_alive = alive;

}

void ClongGray::SetIsShow(bool show) {

is\_show = show;

}

void ClongGray::SetXY(int nx, int ny)

{

x = nx; y = ny;

}

void ClongGray::OnShow()

{

if (is\_alive) {

bmp.SetTopLeft(x, y);

bmp.ShowBitmap();

}

}

}