**mygame.h**

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 OnKeyDown(UINT, UINT, UINT); // 處理鍵盤Down的動作

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int order;

CMovingBitmap pic[4];

};

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);

static void ToBonus(){TimeToBonus=PassMusicTime;}

static int multiplayer,Chip\_Dale,NowLevel; // multiplay 1= 1P ,2= 2P nowplayer 0 =Chip, 1= Dale

static ChipDale \*player[2];

static bool TimeToGo;

static bool IfViewer;

static bool IsBoss;

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CMovingBitmap saved,loading; // 背景圖

CMovingBitmap Health[3],ShowWho[3];

CFrame Frame;

MapManage \*Maps;

GameMapEdit EditPointer; //編輯箭頭

Monster \*AllMonster[ONE\_LEVEL\_MONSTER\_NUM]; // 全部怪物

Object \*AllThrow[CanTakeNum];

int EditMoveMode;

bool show\_save,show\_load;

static int TimeToBonus;

};

class CGameStateBonus : public CGameState {

public:

CGameStateBonus(CGame \*g);

~CGameStateBonus();

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

};

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 CAudio {

public:

//自定義函式

void PauseById(int id);

void SetSpeedByID(unsigned int id,int speed);

void PlayOnSpe(unsigned,bool repeat=false,int time=0);

void Resume\_for\_CGame();

//Functions for Game\_ChipDale

void PlayLevelMusic(int level,bool stop\_and\_play=false); //限於GameStateRun使用，會自動偵測是否該播放BOSS音樂

void PauseLevelMusic(int level); //限於GameStateRun使用，會自動偵測是否該播放BOSS音樂

void StopLevelMusic(int level); //限於GameStateRun使用，背景音樂與BOSS音樂全部stop

void LevelMusicOnit(); //LevelMusic對照Map初始化(只能執行一次，有防呆)

private:

static map<int,int> LevelMusic;

};

class ToolCDC{

public:

ToolCDC();

static void Loading(); //為 ShowProgressBar 的圖片

static void Fadeout(){NowState=1;} //setState =1

static void Fadein(){NowState=4;} //setState =1

static void SaveCDC(); //記錄目前螢幕的 Colors

static int ReturnStage(){return NowState;} //傳回目前該做甚麼事情

static void ToNextState(); //移到下一個state

static void CDDraw();

static void ShowProgressBar(int WhichLevel,int percent);

private:

static COLORREF color[MHEIGHT][MWIDTH];

static CAnimation MovePointer;

static CMovingBitmap loading\_BG,loading\_Bar,loading\_Mask,level,letter[LEVELNUM];

static int NowState;

//0 未設定 ,

//1 下一次要OnShow要save & 鎖人物移動恩螢幕顯示, (淡出)

//2 saved 跟onMove說可以換人物位置 & 鎖人物移動恩螢幕顯示 ,

//3 鎖人物移動恩螢幕顯示

//4 下一次OnShow要save & 鎖人物移動恩螢幕顯示, (淡入)

//5 鎖人物移動恩螢幕顯示

static double alpha; //目前的aplha值

};

class GameMapEdit{

public:

GameMapEdit();

void Loading();

void OnShow(MapManage \*);

void SetOrder(int);

int nowOrder(){return Order;};

void OnMove(MapManage \*map);

void SetWxWy(int setWx,int setWy);

void SetMove(bool,bool,bool,bool,bool,int mode=0);

void SetObject(MapManage \*map); //去wx wy 位置放置obj

int ReturnWX(){return wx;}

int ReturnWY(){return wy;}

void FixXY(MapManage \*); // 修正螢幕位置

private:

int wx,wy;

int Order;

int ThisMoveTimesX,ThisMoveTimesY; // X、Y方向的移動次數

int ContinueMode; // TAB切換微調模式 0 1 2

bool move[4];

CMovingBitmap frame\_pic;

CMovingBitmap frame\_background[OrderSize+MonsterOrderSize];

};

class GameScore{

public:

static GameScore\* Instance();

~GameScore();

void Loading();

void ShowScroe();

void onShow();

bool Switch;

private:

GameScore();

static GameScore Game\_Score;

CMovingBitmap Head[2];

CMovingBitmap F,S;

CMovingBitmap L\_C,L\_D;

CInteger\* score\_p;

};

class CFrame{

public:

CFrame();

void SetWxWy(int wx,int wy,bool shiftmode=false);

virtual void FixXY(MapManage \*); // 修正螢幕位置

virtual void ReFixXY(MapManage \*); // 反修正螢幕位置

virtual void FixMapMove(int fixX,int fixY);

void SetWidthHeight(int tempW,int tempH);

virtual int IfCollision(int twx,int twy,int twidth,int theight);

int IfCollision(int Direct,int passSpeed); //與地圖做運算碰撞

virtual void CollisionReact(int setDirect,CFrame \*which){;}

int ReturnWX(){return wx;} // 回傳螢幕x座標

int ReturnWY(){return wy;} // 回傳螢幕y座標

int ReturnWidth(){return width;}

int ReturnHeight(){return height;}

bool ReturnLR(){return LRflag;}

static int Jump\_Fix;

private:

protected:

int wx,wy; // 框框左上角的螢幕座標 window x,y

int width,height;

bool flag[6]; // 上下左右AB 按鍵旗標

bool LRflag; // 0 左 1 右 標示方向

};

**mapmanage.h**

class MapManage{

public:

MapManage(int WhichLevel);

void OnShow();

void OnShowObject();

void LevelLoading(int WhichLevel,bool NotResetAll=false); //設定關卡 NotResetAll是否重設此關所有變數

int MoveMap(int direct,int MoveOneX=SPEED,int MoveOneY=SPEED);//move 1,2,4,8 ...

void SetMapXY(int,int); //設定 map 顯示位置

void SetObstacle(); //設定障礙物資料

void SetMonster(Monster \*monster[]); //設定怪物資料

void SetObj(int value,int wx,int wy); //wx,wy 螢幕座標

void SetRecord(){recordx=x;recordy=y;} //設定目前地圖左上角為紀錄點

int ReturnNowX(){return x;} //傳回螢幕左上角在地圖的x位置

int ReturnNowY(){return y;} //傳回螢幕左上角在地圖的y位置

int ReturnOBJ(int ox,int oy){return MapObjXY[oy][ox];} //回傳 Obj 的值

void SaveObj(int WhichLevel); //儲存 Obj 資訊

void ShowObstacle(int i,int j); //顯示指定的物件 ps: 給edit用的

void ClearisMoveMap(bool flag); //清除 isMoveMap 紀錄

void Set\_toRecord();

int IfCollision(int wx,int wy,int PicWidth,int PicHeight,bool ignore\_obj\_2=false,bool ignore\_eat=false,bool findNext=false,bool Reverse=false); //計算是否碰撞

int GetRoute(){return Route[y/MHEIGHT][x/MWIDTH];}

int GetRoute(int rx,int ry){return Route[ry][rx];}

void ClearObstacle(int Value); //把目標物件值告訴它 讓他直接去刪除碰撞機制

int FillObstacle(int Value,int setWx,int setWy,bool Visible=true); //把目標物件值告訴它 讓他直接去增加碰撞機制

static int isMoveMap; //地圖移動方向 1,2,4,8 上下左右

static int Teleport;

static int LRMargin;

friend class ChipDale;

private:

CMovingBitmap frame\_obstacle[OrderSize+MonsterOrderSize+UnVisibleOrderSize];

protected:

int Route[MAX\_YN][MAX\_XN]; //地圖路線移動方向 1,2,4,8 方向 16 出生點 32~ 64 ~boss

int MapObjXY[MAX\_OY][MAX\_OX]; //物件座標~顯示讀寫檔用

int Obstacle[MAX\_OY][MAX\_OX]; //障礙物 判斷碰撞用 //第一部分：種類 第二：左上Y 第三 左上X

CMovingBitmap MapSP[MAX\_YN][MAX\_XN];

int x,y; //目前螢幕左上角在地圖的座標

int recordx,recordy; //死亡起始位置

};

**instruction.h**

class AI\_Instructions : public CFrame{

public:

AI\_Instructions();

bool MoveUp(MapManage\* map);

bool MoveDown(MapManage\* map);

bool MoveRight(MapManage\* map);

bool MoveLeft(MapManage\* map);

void Jump();

int FallingDown(MapManage\* map);

protected:

int UpSpeed,MoveSPEED,JumpSPEED;

int LR\_Space;

bool IsJump;

bool NoCollision; // NoCollision = true 不要判斷碰撞

bool NoIgnore\_2,NoIgnore\_eat;

};

class BasicInstructions : public CFrame{

public:

BasicInstructions();

bool MoveRight(MapManage\* map);

bool MoveLeft(MapManage\* map);

bool Jump();

void Squat();

int FallingDown(MapManage\* map);

virtual ChipDale\* getPartner(){return NULL;}

virtual int CollisionChipDale(int Direct,int passSpeed,int Squeezemode=1){return false;}

virtual void TriggerObj(int Derict){TRACE("TriggerObj error\n");} // Derict吃花的方向

virtual int SetNowTaken(int Value){return -1;} //設定NowTaken

virtual ChipDale\* Instance(){return NULL;}

protected:

int UpSpeed,time\_jump; // time\_jump 避免二段跳

bool IsJump,IsRun,IsLessCollision; // isLessCollison = true 不要判斷地板2的碰撞(下跳觸發)

int ani\_jump\_count; // animation 跳轉蹲的計數器

int Reduce\_UP\_VELOCITY;

static bool canMoveMapX,canMoveMapY;

};

**ChipDale.h**

class ChipDale : public BasicInstructions{

public:

ChipDale(int isDale);

static void Loading();

void OnMove();

void GodMove();

void OnShow();

ChipDale\* Instance(){return this;}

int ReturnHideComplete(){return (animation[IsDale][1][4][LRflag].GetCurrentBitmapNumber()>=4)&&NowAct==4;}

int ReturnNowAct(){return NowAct;}

bool ReturnIsHurting(){return Hurt;}

int ReturnLastAct(){return LastAct;}

int ReturnIsTaken(){if(NowTaken>0)return 1;else return 0;}

int ReturnTakenByPartner(){if(Partner!=NULL&&Partner->NowTaken==ChipDale\_taken)return 1 ;else return 0;}

int ReturnHealth(){return Health;}

int ReturnLife(){return Life;}

int ReturnInvincible(){return Invincible;}

int ReturnIsDale(){return IsDale;}

int CollisionChipDale(int Direct,int passSpeed,int Squeezemode=1);

Object\* ReturnNowTakeObj(){return NowTakeObj;}

ChipDale\* getPartner(){return Partner;}

void setFlag(bool value,bool,bool,bool,bool,bool,bool); //值 上下左右AB

void SetState(); //設定Act LRdirect

void SetPartner(ChipDale \*setPlayer){Partner = setPlayer;}

void SetInvincible(int value){Invincible=value;} //設定Invincible 1無敵 0正常

void ReleaseNowTakeObj();

void FixSelf\_Onto\_ObjectTop();

void InitialWidthHeight(); //初始化高度

void Reset(int Wx,int Wy,bool LR=1,bool FullHealth=1);

void ResetScore(){Score\_Flower=Score\_Star=0;}

void Dead();

void Lock(bool value = true ){LOCK=value;}

void GetHurt();

void Faint();

void GodMode();

virtual int SetNowTaken(int Value); //設定NowTaken

virtual void TriggerObj(int Derict); //Derict吃花的方向

int Score\_Flower;

int Score\_Star;

static Object \*\*CanThrow;

static MapManage \*Maps;

static Monster \*\*AllMonster;

friend class BasicInstructions;

friend class ChipDale\_Dead;

friend class ChipDale\_Resurrect;

friend class ChipDale\_Taken;

private:

static CAnimation animation[2][2][ACTION\_NUM][2]; // 動作~方向

static CAnimation ani\_sweat[2],ani\_dizzy,ani\_god[2];

int IsDale;

int ani\_sweat\_x;

int ani\_sweat\_wy;

int ani\_sweat\_count;

int Health;

int Life;

int NowAct,LastAct; // Act 0站 1走 2跳 3丟 4蹲 5受傷 6暈眩

int NowTaken,LastTaken; // 0空手 1拿普通東西 6蘋果... (<0 作為再次拿取延遲器)

int freeze; // 丟東西的冷卻時間

bool Hurt; // 是否受傷

int IsFaint; // 是否暈眩 ,counter合一

bool ani\_Hide\_freeze\_jump; // 控制HIDE動畫未完成不能下跳，設定在CHIPDALE的動畫切換

int Last\_flag[6]; // 上一次按鍵的旗標，搭配flag做出放開按鍵判斷

int Invincible; // 1無敵 0正常 (>1作為ONSHOW的COUNTER)

Object \*NowTakeObj;

ChipDale \*Partner;

bool IsGod;

bool LOCK; // 鎖setFlag

bool Alive; // 鎖除了setFlag以外的函式

};

**object.h**

class Object : public AI\_Instructions{

public:

Object();

virtual ~Object(){;}

virtual void OnShow(MapManage\* map){;}

virtual void OnMove(MapManage\* map){;}

virtual void Throw(int setDirect){;}

int ReturnObjValue(){return ThisObjValue;}

int ReturnNowAct(){return NowAct;}

void SetNowAct(int setNowAct) {NowAct = setNowAct;}

void RecoverObj(ChipDale \*player); //重新拾獲時需呼叫

void CollisionMonster(Monster \*\*monster);

void CollisionChipDale(ChipDale \*player);

protected:

int CanAttackMode; // 攻擊對象 0 任何人 1 除了主人(預設值),2 None 3 只攻擊怪物

int Direct,LRflag,ThisObjValue,NowAct;

//ThisObjValue用來記錄刪除obstacle的編號

//NowAct =0 停屍間 1 拿在手上 2運動狀態 3放在地上 4打完怪物

ChipDale \*Owener;

};

class ChipDale\_Dead:public Object{

public:

ChipDale\_Dead(ChipDale\* chip);

~ChipDale\_Dead();

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

friend class ChipDale\_Resurrect;

private:

int isDale;

int Shine\_Count;

static CMovingBitmap frame\_pic[2][2];

int Wait\_A\_Minute;

};

class ChipDale\_Resurrect:public Object{

public:

ChipDale\_Resurrect(ChipDale\_Dead\* chip);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int isDale;

int timeCount;

static CMovingBitmap frame\_pic[2][2];

};

class ChipDale\_Taken:public Object{

public:

ChipDale\_Taken(ChipDale\* chip);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

private:

int isDale;

static CAnimation animation[2][2];

};

class Star : public Object{

public:

Star(int setWx,int setWy);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int lastUpSpeed;

static CMovingBitmap frame\_pic;

};

class Cheese : public Object{

public:

Cheese(int setWx,int setWy);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int lastUpSpeed;

static CMovingBitmap frame\_pic;

};

class Explosion : public Object{

public:

Explosion(int setOx,int setOy,int setChangeWhat,ChipDale \*player); //ChangeWhat 填代碼 (跟顯示物件代碼一樣)

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int ChangeWhat;

bool IfNeedReFix;

static CAnimation frame\_pic;

};

class Angel : public Object{

public:

Angel(int setWx,int setWy,ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int NowAttack;

bool OutLRflag;

const int TheBeeMaxSpeed;

static CAnimation frame\_pic[2];

static Monster \*\*monster;

};

class Apple : public Object{

public:

Apple(ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

private:

static CMovingBitmap frame\_pic[2];

};

class Ball : public Object{

public:

Ball(ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

private:

int Rebound\_times;

static CMovingBitmap frame\_pic[2];

};

class Craft : public Object{

public:

Craft(ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

void CollisionReact(int setDirect,CFrame \*which);

~Craft(){TRACE("Craft destuctor run\n");}

private:

static CMovingBitmap frame\_pic[2];

};

class Stone : public Object{

public:

Stone(ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

void CollisionReact(int setDirect,CFrame \*which);

bool MoveRight(MapManage\* map);

bool MoveLeft(MapManage\* map);

void ReBound(MapManage\* map);

private:

bool HideThrow;

static CMovingBitmap frame\_pic[2];

int ReboundLR; // 0 無反彈 1 反彈右邊 -1反彈左邊

int ReboundSpeed;

};

class Greedy : public Object{

public:

Greedy(int setWx,int setWy,Object \*setRemoveObj);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int PassWx;

bool showMouse;

static CAnimation frame\_pic[2];

static CMovingBitmap frame\_hole;

Object \*CheeseObj;

};

**monster.h**

class Monster : public AI\_Instructions{

public:

Monster();

virtual ~Monster(){;}

void SetNowAct(int Value){NowAct = Value;}

int ReturnNowAct(){return NowAct;}

bool ReturnCanTrace(){return CanTrace;}

virtual void CollisionChipDale(ChipDale \*player);

virtual void OnShow(MapManage \*map){;}

virtual bool OnMove(MapManage \*map,ChipDale \*\*player){return false;}

virtual void SetMonster(MapManage \*map,int SetOx,int SetOy){;}

virtual bool KillMonster(int Direct){return false;}

ChipDale\* Detect(ChipDale \*\*player,int \*WLength,int \*HLength,int WRange,int HRange,bool IfTraceInvincible=false);

friend class ChipDale;

protected:

int LR\_flag,NowAct; //NowAct 正常介於1-100 100UP作為彈飛動畫

int lastWy,wait;

int Health;

bool CanTrace; //是否可以被跟蹤 給其他OBJ用 (ex: Object::Angel)

};

class bullet : public Monster{

public:

bullet();

virtual void OnShow(MapManage \*map){;}

bool OnMove(MapManage \*map);

int Collision(ChipDale \*player){return player->IfCollision(wx,wy,width,height);}

protected:

int MoveLR,MoveUD;

};

class MachineDog : public Monster{

public:

MachineDog(MapManage \*map,int SetOx,int SetOy);

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

bool KillMonster(int Direct);

private:

static CAnimation frame\_monster[2][2]; //NowAct LRflag

};

class Cactus : public Monster{

public:

Cactus(MapManage \*map,int SetOx,int SetOy);

static void Loading();

bool OnMove(MapManage \*map,ChipDale \*\*player);

private:

static CMovingBitmap frame\_monster;

};

class Electric : public Monster{

public:

Electric(MapManage \*map,int SetWx,int SetWy,int SetWireOLength);

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

int ElectricCollision(ChipDale \*player){return player->IfCollision(wx,wy,width,height);}

private:

int MoveLength;

int WireLength;

const int ChangeSpeed,MaxMoveSpeed;

static CMovingBitmap frame\_monster[2];

};

class Wire : public Monster{

public:

Wire(MapManage \*map,int SetOx,int SetOy,int SetWireLength);

~Wire(){delete(real\_monster);}

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

void CollisionChipDale(ChipDale \*player);

void FixMapMove(int fixX,int fixY);

private:

int WireLength;

Electric \*real\_monster;

};

class Mouse : public Monster{

public:

Mouse(MapManage \*map,int SetOx,int SetOy);

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

bool KillMonster(int Direct);

private:

ChipDale \*tracePlayer;

static CAnimation frame\_monster[3][2]; //NowAct LRflag

};

class Wasp : public Monster{

public:

Wasp(MapManage \*map,int SetOx,int SetOy);

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

bool KillMonster(int Direct);

private:

int tLRMoveSPEED,tDownMoveSPEED;

static CAnimation frame\_monster[2];

};

class Lighting : public bullet{

public:

Lighting(MapManage \*map,int SetWx,int SetWy,int LR,int UD);

static void Loading();

void OnShow(MapManage \*map);

private:

static CMovingBitmap frame\_monster;

};

class Centipedelimbs : public Monster{

public:

Centipedelimbs(MapManage \*map,int SetWx,int SetWy,int SetSelect);

~Centipedelimbs();

static void Loading();

void OnShow(MapManage \*map,int countFlicker);

bool OnMove(MapManage \*map,ChipDale \*\*player);

int CentipedelimbsCollision(ChipDale \*player);

private:

static CAnimation frame\_monster[3];

bullet \*bullets[CentipedeLightingNum];

int Select;

const int TotalBullTime,ReleaseBullTime,BulletMaxSpeed,WRandBullet;

};

class Centipede : public Monster{

public:

Centipede(MapManage \*map,int SetOx,int SetOy);

~Centipede();

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

void CollisionChipDale(ChipDale \*player);

void FixMapMove(int fixX,int fixY);

int IfCollision(int twx,int twy,int twidth,int theight){return real\_monster[0]->IfCollision(twx,twy,twidth,theight);}

bool KillMonster(int Direct);

private:

Centipedelimbs \*real\_monster[3];

int countFlicker;

static CAnimation frame\_monster;

};