Sword Art Offline\_课程设计报告

# 组员个人信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 手机号 | 邮箱 | 院系 |
| 141270010 | 吉也 | 13022593119 | 2731036958@qq.com | 工程管理学院 |
| 141279010 | 胡子韬 | 18351885813 | hzt58@qq.com | 工程管理学院 |
| 141270013 | 雷天赫 | 18362918221 | 141270013@smail.nju.edu.cn | 工程管理学院 |

# 程序功能及组员分工

Inspired by anime Sword Art Online. We intend to rebuild the fantastic virtual world in 2D.

This game is built in respect for the limitless virtual reality world composed by Heathcliff.

功能预期：

基于SAO（刀剑神域）理念的火柴人（动作XD）角色扮演类游戏（RPG）

游戏目前实现了unlimited模式，其中包含基本战斗、基本移动、不同地图间的移动

画面流畅，代码结构清晰，少BUG

关键词：

SAO、桐人、横版、火柴人、RPG、同人游戏

我们的分工：

吉也：美术总监——在动漫的基础上力图还原主角动作及技能（以火柴人形式）

胡子韬：美术编辑——设计敌人动作（同上），还原地图背景

雷天赫：程序猿——代码编写

# 程序框架图

main()

welcomeInit()

gameExit()

authorInfo ()

(in plan)

SAOMode()

(inplan)

unlimitedMode()

(developing)

游戏准备：

1. HP\_BAR、背景图片加载
2. Bgm播放
3. 开始批量绘图

主循环

主循环内容：

While（1）

{

1.玩家和敌人存活确认

2.根据玩家位置判断背景移动

3.检测键盘输入（\_kbhit）

J||K||L：技能flag立起。

A或D：移动flag立起；方向改变；移动一次；根据按键长短输出对应动作图片。

W: 检测人物位置决定是否传送，传送到哪，并加载新地图。

SPACE：跳跃flag立起

ESC：gameExit()

4.检测是否有跳跃flag

True：根据jumpstate做出纵向位移

False：无操作

5.检测是否有技能flag

True：根据dir、skillType、combo、skillState判断输出的技能姿势和特效

False：无操作

6.是否静止

True：输出kirito静止姿势

False：无操作

7.批量绘图输出

8.Sleep(10)

9.清空除背景外的图像

}

# 数据结构

主框架

typedef class MAINFRAME

{

public: /\*构造、析构函数\*/

MAINFRAME();

~MAINFRAME();

public: /\*功能\*/

bool welcomeInit(); //初始化欢迎界面

//void saoMode(); //SAO模式(未实现)

void unlimitedMode(); //无限模式

void gameExit(); //退出游戏

void bgm(int); //播放bgm

void sound(int); //播放音效

void setOriginx(int); //重设绘图原点

int getOriginx(); //获取绘图原点

void sceneChange(PLAYER\*, int,IMAGE\*); //场景转换

void screenMove(int,int); //屏幕适应性移动

void hpUI(int\*, int\*); //kiritoのHPのUI

void otherUI(PLAYER\*); //其他的UI

void achievement(PLAYER\*); //成就显示

void credit(IMAGE\*,IMAGE\*,IMAGE\*,int\*); //制作人员

void roadBar(IMAGE\*); //路标

void enemyHpUI(int\*, int\*,int\*,int\*); //敌人のHP

void bossHpUI(int\*, int\*, int\*, int\*); //BOSSのHP

void loadOrNew(); //选择读档或者开始新游戏(未实现)

void stillput(bool,int,int,int, IMAGE\*,int); //静止时图片加载

int fileRead(); //读取文件

static void M\_putimg(int,int,IMAGE\*,int,int,int);//绘制图片（背景透明&透明度可调）

static void HPCAO\_putimg(int, int, IMAGE\*, int, int, int);

static void copy\_img(IMAGE\*, IMAGE\*); //复制图片，机巧图片不会受伤

protected:

int originx; //绘图原点

int floor; //当前层数

int highfloor; //所达最高层数

int mobslaynum; //所消灭怪物数量

int deathtime; //死亡次数

int room; //房间编号

int leftlimit; //房间左边界

int rightlimit; //房间右边界

bool BGM; //是否播放BGM

bool SOUND; //是否播放音效（MAINFRAME的音效）

PLAYER kirito; //玩家数据库

}mainFrame;

玩家

typedef class PLAYER

{

public: /\*构造析构函数\*/

PLAYER();

~PLAYER();

public:

int getX(); //获取位置

int getY();

int getMovespd(); //获取移动速度

int getAttack(); //获取攻击力

float getHp(); //获取当前hp

int getSkill(); //获取当前释放技能编号

int getCombo(); //获取当前连击值

void meleeAttack(int\*, int\*, PLAYER\*, IMAGE\*, IMAGE\*, IMAGE\*, IMAGE\*, int); //普通の攻击

void sonicLeap(int\*, int\*, PLAYER\*, IMAGE\*, IMAGE\*, IMAGE\*, IMAGE\*, int); //音速冲击

void horizontalSquare(int\*, int\*, PLAYER\*, IMAGE\*, IMAGE\*, IMAGE\*, IMAGE\*, int);//水平四方斩

void damage(int); //受到伤害

void jump(); //跳跃

void running(IMAGE\*, int,PLAYER\*); //跑动

void moveX(int\*,int\*); //基本移(hua)动

void startJump(); //跳起

void useSkill(int); //释放技能

bool stillJudge(); //判断是否静止

bool jumpJudge(); //判断是否跳跃

bool runJudge(); //判断是否跑动

bool coolingJudge(); //判断是否处于冷却

bool getDir(); //获取方向

int getStill(); //获取still值

void restill(); //恢复静止

int getSkillState(); //获取技能状态

int getSkillType(); //获取技能类型

int getRunState(); //获取跑动状态

void skillEffect(IMAGE\*,IMAGE\*,IMAGE\*,IMAGE\*,PLAYER\*,int); //给技能加特技

void startRun(); //开始跑动

void coolingDown(); //冷却恢复

void skillStateMove(); //计算skillState

float getMaxHp(); //获取最大hp

void AIFind(int\*); //enemy追踪玩家

void AIAttack(PLAYER\*,IMAGE\*); //enemy攻击玩家

int getLev(); //获取等级

void levelUp(); //升级啦hhhhh

void statRefresh(); //根据等级刷新数值

int getExp(); //获取当前exp

void expAdd(int); //经验增加

void levelUpCheck(); //检测是否升级

void hpRege(); //被动HP回复

void respawn(); //复活，经验损失10%

int getType(); //获取种类

int getRun(); //获取run值

void bossAttack(int\*, int\*, PLAYER\*, IMAGE\*, IMAGE\*, int);//BOSS的攻击

bool getSound(); //获取sound值

//初始化用函数

void teleport(int, int);//用于玩家传送

void setDir(bool); //设置方向

void setHp(float); //设置Hp值

void setMaxHp(float); //设置MaxHp值

void setAttack(int); //设置attack值

void setSpd(int); //设置移动速度

void setCombo(int); //设置连击情况

void setSkillState(int);//设置技能释放状态

void setSkillType(int); //设置技能类型

void setStill(int); //使静止

void setRunState(int); //设置跑动阶段

void setRun(int); //设置run值

void setLevel(int); //设置等级

void setExp(int); //设置经验值

void setType(int); //设置种类

void soundEffect(int); //声效

void setSound(bool); //设定音效开关状态

protected: /\*\*\*\*玩家属性\*\*\*\*/

float hp; //当前生命值

float maxhp; //最大生命值

int exp; //经验值

int level; //等级

float hpregen; //每帧战斗回复hp（注:FPS约100）

int movespd; //每次移动像素数

int attack; //攻击力

int x; //位置

int y;

bool dir; //0->左 1->右

int combo; //连击状态

int jumpState; //跳跃过程

int runState; //跑动过程

int run; //跑动核心参数

int coolDown; //技能冷却判断

int skillState; //技能释放过程

int skillType; //当前释放的技能编号

int still; //san值

bool isRun; //跑动

bool isJump; //跳跃

bool cooling; //是否处于攻击状态

int type; //0玩家；1-9普通敌人；10BOSS

bool SOUND; //音效开关

}kirito,enemy;

剩余项目：

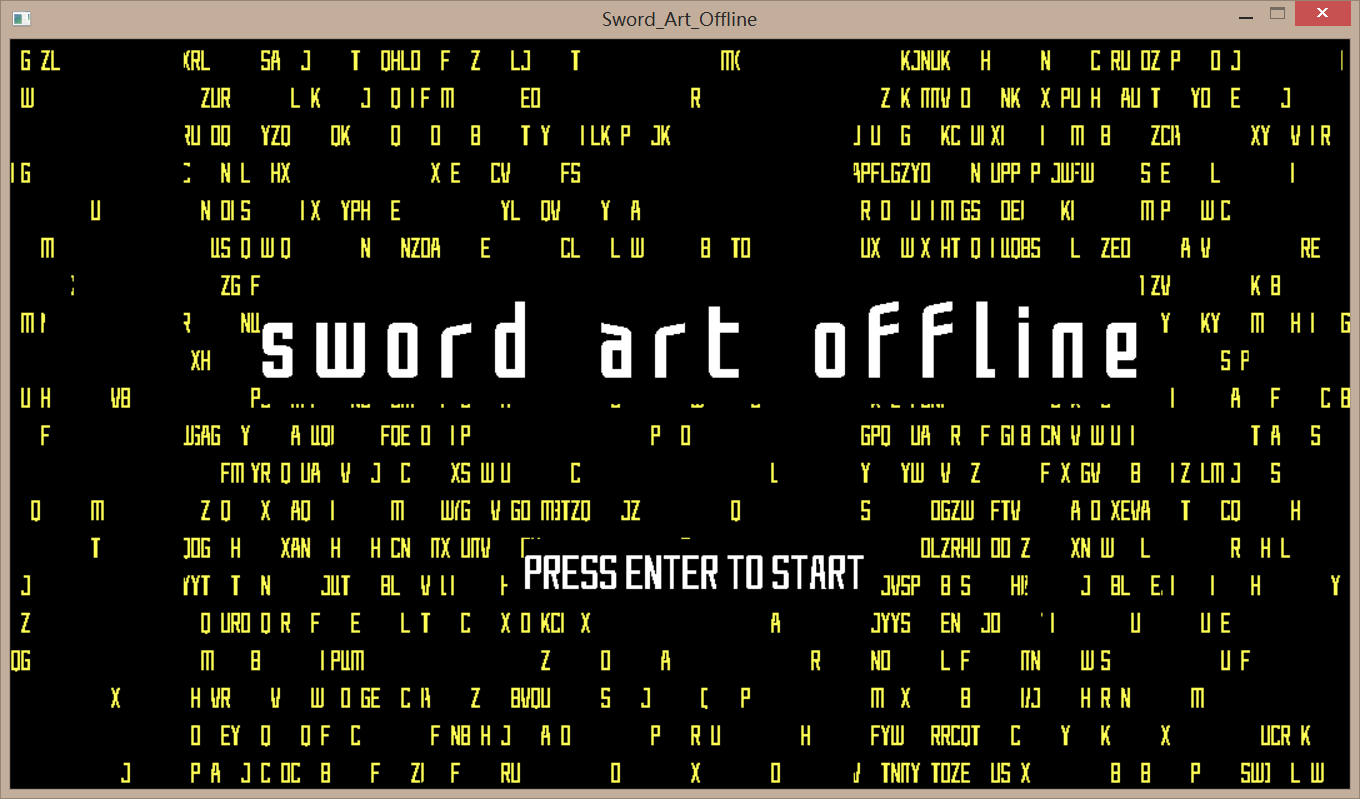
1. 主要模块的功能及工作流程图

# 主要模块的功能及工作流程图

# 调试过程

1. 因为游戏是根据动漫原作创作的，所以我们一直在努力还原原作的艺术效果。在欢迎界面方面，我们没有采用一般的多选项主菜单，而是制作了还原动漫开头（OP）中的背景，并提供了开始游戏的选项。

动漫原场景&我们的界面



如果把EasyX中提到的获取鼠标信息的函数放在主循环中，当没有鼠标信息（鼠标不动）的时候会中断循环的进行，这对于欢迎界面是致命的，因为背景字母的输出依赖于主循环的不断执行。最后的解决方案是，换用一个更加直接获取鼠标信息的函数（GetCursorPos(&pt); ScreenToClient(hw, &pt);），于是避免了主循环的走走停停的情况。

1. 在人物贴图方面，我们因为避免使得工作量过大而采用了火柴人（然而工作量还是不小），实现了玩家基本的移动（依照第二集中男主的跑步姿势绘制），技能的动作（依照男主分散在各集的三个攻击动作绘制）和特效（依照剑的挥动绘制），敌人的攻击（自制）与被攻击（自制）等等的贴图，营造了较为连贯的动作感。调试过程中，每个技能的特效贴图位置需要一遍遍的修正才能达到理想的效果，每个技能所参考的skillstate值的变动需要与技能相匹配，这需要大量的时间进行调试。开始的时候，普通攻击的特效总是会闪两次，出问题的应该是skillstate的变化或者判定，但是怎么也找不到出错的位置（代码已经开始变长啦……），但是这个问题在程序结构大改之后消失了！最终的效果还不错哦~
2. 在地图背景方面，除了根据房间判定进行地图的绘制外，只需要简单的调试确定传送判定的位置，确定输出地图元素的位置就好啦，还是比较简单的。
3. 主结构方面，需要协调各个功能的先后与调用，每帧进行存活判定、计算冷却时间、技能释放过程、still值恢复等等，尤其是依靠指针进行比较复杂的信息传递，也是消耗了大量时间。
4. 后期完善的过程中，加了一些增强游戏性的元素。比如UI显示，操作提示，路标，成就屋，音效开关，复活，传送。并调整了怪物难度的变化依据（由玩家等级更换为当前楼层数），也就是说楼层越高怪物越难对付，反之亦然。并设置在击杀BOSS后传送到制作者房间，这个房间还是很带感的。
5. 作为一个新手，EasyX还是比较友好的，但是，在进行较为复杂的动作绘制时总觉得应该换个更好的库效果来的更好……曾经考虑过换HGE，但因为使用EasyX的这个版本已经做了很久了，就没有放弃掉。

# 课程设计过程中的学习体会

1. 首先感谢学长学姐们探索到了这个对新人真的十分友好的EasyX，可以说，有EasyX才有了我们的这个作品=w=。
2. 游戏是按照自己和室友对动漫的热爱去构思的，也许有点只顾个人的喜好，但我们也想到游戏所面向的大家中一定也有很多人喜欢动漫啦，并且，希望这个游戏能够承载我们对于动漫和编程的热爱，并将其传递给更多的人。
3. 感受到了敏捷开发的正确性，一步一步写，慢慢完善程序的功能，这样做目标更加明确，化难为简，而且程序结构清晰。
4. 坚持到底，想做的事情就一定能成功。一直梦想着能做出一个酷炫的游戏，但是一开始要用EasyX做动作RPG其实我是拒绝的，可手头没有更理想的库，也就决定试一试。游戏的内容和背景源于很喜欢的动漫，做的时候也不觉得很枯燥，BGM真的是反复的听（哈哈），到现在能做出这样的效果，感觉真的很欣慰。这也许就是工科的乐趣所在啦，做项目的时候可能感觉很累，但是做出结果，而且效果不错的时候却又高兴得一比。写完这个程序，也是为我们积累了宝贵的经验，并为我们日后的其它计算机语言的学习打下了更好的基础，能有这样的机会和小组的成员一起完成这个游戏，感觉自己真的很幸运！

# 完整源程序清单

Mainframe.h

#pragma once

#include <iostream>

#include <stdlib.h>

#include <stdio.h>

#include <graphics.h>

#include <time.h>

#include <conio.h>

#include "player.h"

using namespace std;

typedef class MAINFRAME

{

public: /\*构造、析构函数\*/

MAINFRAME();

~MAINFRAME();

public: /\*功能\*/

bool welcomeInit(); //初始化欢迎界面

//void saoMode(); //SAO模式(未实现)

void unlimitedMode(); //无限模式

void gameExit(); //退出游戏

void bgm(int); //播放bgm

void sound(int); //播放音效

void setOriginx(int); //重设绘图原点

int getOriginx(); //获取绘图原点

void sceneChange(PLAYER\*, int,IMAGE\*); //场景转换

void screenMove(int,int); //屏幕适应性移动

void hpUI(int\*, int\*); //kiritoのHPのUI

void otherUI(PLAYER\*); //其他的UI

void achievement(PLAYER\*); //成就显示

void credit(IMAGE\*,IMAGE\*,IMAGE\*,int\*); //制作人员

void roadBar(IMAGE\*); //路标

void enemyHpUI(int\*, int\*,int\*,int\*); //敌人のHP

void bossHpUI(int\*, int\*, int\*, int\*); //BOSSのHP

void loadOrNew(); //选择读档或者开始新游戏(未实现)

void stillput(bool,int,int,int, IMAGE\*,int); //静止时图片加载

int fileRead(); //读取文件

static void M\_putimg(int,int,IMAGE\*,int,int,int);//绘制图片（背景透明&透明度可调）

static void HPCAO\_putimg(int, int, IMAGE\*, int, int, int);

static void copy\_img(IMAGE\*, IMAGE\*); //复制图片，机巧图片不会受伤

protected:

int originx; //绘图原点

int floor; //当前层数

int highfloor; //所达最高层数

int mobslaynum; //所消灭怪物数量

int deathtime; //死亡次数

int room; //房间编号

int leftlimit; //房间左边界

int rightlimit; //房间右边界

bool BGM; //是否播放BGM

bool SOUND; //是否播放音效（MAINFRAME内的音效）

PLAYER kirito; //玩家数据库

}mainFrame;

player.h

#pragma once

#include <graphics.h>

using namespace std;

typedef class PLAYER

{

public: /\*构造析构函数\*/

PLAYER(); //默认是空

~PLAYER();

public:

int getX(); //获取位置

int getY();

int getMovespd(); //获取移动速度

int getAttack(); //获取攻击力

float getHp(); //获取当前hp

int getSkill(); //获取当前释放技能编号

int getCombo(); //获取当前连击值

void meleeAttack(int\*, int\*, PLAYER\*, IMAGE\*, IMAGE\*, IMAGE\*, IMAGE\*, int); //普通の攻击

void sonicLeap(int\*, int\*, PLAYER\*, IMAGE\*, IMAGE\*, IMAGE\*, IMAGE\*, int); //音速冲击

void horizontalSquare(int\*, int\*, PLAYER\*, IMAGE\*, IMAGE\*, IMAGE\*, IMAGE\*, int);//水平四方斩

void damage(int); //受到伤害

void jump(); //跳跃

void running(IMAGE\*, int,PLAYER\*); //跑动

void moveX(int\*,int\*); //基本移(hua)动

void startJump(); //跳起

void useSkill(int); //释放技能

bool stillJudge(); //判断是否静止

bool jumpJudge(); //判断是否跳跃

bool runJudge(); //判断是否跑动

bool coolingJudge(); //判断是否处于冷却

bool getDir(); //获取方向

int getStill(); //获取still值

void restill(); //恢复静止

int getSkillState(); //获取技能状态

int getSkillType(); //获取技能类型

int getRunState(); //获取跑动状态

void skillEffect(IMAGE\*,IMAGE\*,IMAGE\*,IMAGE\*,PLAYER\*,int); //给技能加特技

void startRun(); //开始跑动

void coolingDown(); //冷却恢复

void skillStateMove(); //计算skillState

float getMaxHp(); //获取最大hp

void AIFind(int\*); //enemy追踪玩家

void AIAttack(PLAYER\*,IMAGE\*); //enemy攻击玩家

int getLev(); //获取等级

void levelUp(); //升级啦hhhhh

void statRefresh(); //根据等级刷新数值

int getExp(); //获取当前exp

void expAdd(int); //经验增加

void levelUpCheck(); //检测是否升级

void hpRege(); //被动HP回复

void respawn(); //复活，经验损失10%

int getType(); //获取种类

int getRun(); //获取run值

void bossAttack(int\*, int\*, PLAYER\*, IMAGE\*, IMAGE\*, int);//BOSS的攻击

bool getSound(); //获取sound值

//初始化用函数

void teleport(int, int);//用于玩家传送

void setDir(bool); //设置方向

void setHp(float); //设置Hp值

void setMaxHp(float); //设置MaxHp值

void setAttack(int); //设置attack值

void setSpd(int); //设置移动速度

void setCombo(int); //设置连击情况

void setSkillState(int);//设置技能释放状态

void setSkillType(int); //设置技能类型

void setStill(int); //使静止

void setRunState(int); //设置跑动阶段

void setRun(int); //设置run值

void setLevel(int); //设置等级

void setExp(int); //设置经验值

void setType(int); //设置种类

void soundEffect(int); //声效

void setSound(bool); //设定音效开关状态

protected: /\*\*\*\*玩家属性\*\*\*\*/

float hp; //当前生命值

float maxhp; //最大生命值

int exp; //经验值

int level; //等级

float hpregen; //每帧战斗回复hp（注:FPS约100）

int movespd; //每次移动像素数

int attack; //攻击力

int x; //位置

int y;

bool dir; //0->左 1->右

int combo; //连击状态

int jumpState; //跳跃过程

int runState; //跑动过程

int run; //跑动核心参数

int coolDown; //技能冷却判断

int skillState; //技能释放过程

int skillType; //当前释放的技能编号

int still; //san值

bool isRun; //跑动

bool isJump; //跳跃

bool cooling; //是否处于攻击状态

int type; //0玩家；1-9普通敌人；10BOSS

bool SOUND; //音效开关

}kirito,enemy;

Mainframe.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "mainframe.h"

#include "player.h"

#include <graphics.h>

#include <conio.h>

#include <windows.h>

#include <mmsystem.h>

#include "Vfw.h"

#pragma comment (lib, "winmm.lib")

#pragma comment (lib, "Vfw32.lib")

#define INIT\_ORIGINX 0

#define INIT\_FLOOR 1

#define INIT\_ROOM 1

#define INIT\_MOBSLAYNUM 0

#define INIT\_DEATHTIME 0

#define INIT\_LEFTLIMIT 0

#define INIT\_RIGHTLIMIT 3216

#define INIT\_BGM true

#define INIT\_SOUND true

MAINFRAME::MAINFRAME()

{

originx = INIT\_ORIGINX;

floor = INIT\_FLOOR;

highfloor = INIT\_FLOOR;

mobslaynum = INIT\_MOBSLAYNUM;

deathtime = INIT\_DEATHTIME;

room = INIT\_ROOM;

leftlimit = INIT\_LEFTLIMIT;

rightlimit = INIT\_RIGHTLIMIT;

BGM = INIT\_BGM;

SOUND = INIT\_SOUND;

}

MAINFRAME::~MAINFRAME() {}

#define KEY\_DOWN(vk\_c) (GetAsyncKeyState(vk\_c)&0x8000?1:0) //检测按键和鼠标点击

#define maxx 1072 //初始化屏幕宽高

#define maxy 600

#define inter 10 //初始化Sleep的间隔

void main()

{

MAINFRAME mainFrame;

mainFrame.welcomeInit();

}

void headLine(char \*SAO, char \*ETS,char \*blank,bool flagstart,bool blink)

{

settextstyle(112, 30, \_T("SAO UI")); // 设置大标题字体

settextcolor(WHITE);

outtextxy(200, 180, SAO);

if (blink == 1 || flagstart == 1){

if (flagstart == 1)settextcolor(YELLOW);

settextstyle(50, 13, \_T("SAO UI")); //设置ETS字体

outtextxy(411, 400, ETS);

}

else{

settextstyle(50, 13, \_T("SAO UI"));

outtextxy(411, 400, blank);

} //实现ETS的闪烁

settextcolor(YELLOW);

setlinecolor(BLACK);

settextstyle(30, 8, \_T("SAO UI")); // 重新设置为背景字体

}

void clearline(int i) //实现背景空白行扫描

{

line(i, 0, i, 599);

line(i + 536, 0, i + 536, 599);

line(i - 536, 0, i - 536, 599);

}

void M\_clear(POINT pt, IMAGE \*bk, IMAGE pic)//pt上一个动作图片的输出坐标，bk背景图片

{

IMAGE clear;

SetWorkingImage(bk);//设定当前的绘图设备为背景图片

getimage(&clear, pt.x, pt.y, pic.getwidth(), pic.getheight());//获取图像

SetWorkingImage();//设定回默认绘图窗口

putimage(pt.x, pt.y, &clear);//输出

}

void MAINFRAME::stillput(bool dir,int x,int y,int ox, IMAGE \*player,int t)

{

switch (t){

case 0:{

switch (dir){

case 0:{

loadimage(player, "pic/lstill.jpg",180,185,true);

mainFrame::M\_putimg(x, y, player, WHITE, 100, ox);

}break;

case 1:{

loadimage(player, "pic/rstill.jpg", 180, 185, true);

mainFrame::M\_putimg(x, y, player, WHITE, 100, ox);

}break;

}

}break;

case 9:{

switch (dir){

case 0:{

loadimage(player, "pic/les.jpg", 180, 185, true);

mainFrame::M\_putimg(x, y,player, WHITE, 100, ox);

}break;

case 1:{

loadimage(player, "pic/res.jpg", 180, 185, true);

mainFrame::M\_putimg(x, y, player, WHITE, 100, ox);

}break;

}

}break;

case 10:{

switch (dir){

case 0:{

loadimage(player, "pic/gleam eyes(lstill).jpg", 425, 425, true);

mainFrame::M\_putimg(x, y-240, player, WHITE, 100, ox);

}break;

case 1:{

loadimage(player, "pic/gleam eyes(rstill).jpg", 425, 425, true);

mainFrame::M\_putimg(x, y-240, player, WHITE, 100, ox);

}break;

}

}break;

}

}

int MAINFRAME::fileRead()

{

FILE \*pf = NULL; //文件指针

int filelen = 0;

int i = 0;

char \*buf=NULL;

pf = fopen("text/greet.txt", "r"); //以只读方式打开文件

if (pf == NULL)

{

return 0;

}

else

{

//获得文件长度

fseek(pf, 0, SEEK\_END); //文件指针移到末尾

filelen = ftell(pf); //获得文件当前指针位置，即为文件长度

rewind(pf); //将文件指针移到开头，准备读取

buf = (char\*)malloc(filelen + 1); //新建缓冲区，存储独处的数据

//将缓冲区的数据设置为0

for (i = 0; i<filelen + 1; i++)

buf[i] = 0;

//读取文件

fread(buf, filelen, 1, pf);

//关闭文件

fclose(pf);

//buf中即为要读出的数据

outtextxy(478,200,buf); //输出一下数据，你可以随便怎么用

free(buf); //最后记得要释放

}

return 1;

}

void MAINFRAME::copy\_img(IMAGE\* img1, IMAGE\* img2)

{

//copy img2 to img1

IMAGE\* now\_working = GetWorkingImage();

Resize(img1, img2->getwidth(), img2->getheight());

SetWorkingImage(img2);

getimage(img1, 0, 0, img1->getwidth(), img1->getheight());

SetWorkingImage(now\_working);

}

int MAINFRAME::getOriginx()

{

return originx;

}

void MAINFRAME::setOriginx(int num)

{

originx = num;

}

void MAINFRAME::screenMove(int x, int spd)

{

if (x > (700 - originx)){ //右移动边界

originx -= abs(spd); //屏幕区域位置改变

}

if (x < (200 - originx)){ //左移动边界

originx +=abs(spd);

}

if (originx > 0)originx = 0; //限制originx>=0

if (originx < -(rightlimit - 1072))originx = -(rightlimit - 1072); //限制originx<=2144

setorigin(originx, 0); //重设原点

}

bool MAINFRAME::welcomeInit()

{

bool flagstart = false;

bool flagsound = true;

bool blink = true;

int tic = 0;

char blank[47] = " ";

HWND hw = GetHWnd();

// 设置随机函数种子

srand((unsigned)time(NULL));

// 初始化图形模式

initgraph(maxx, maxy);

POINT pt; //鼠标位置

int x, y;

char c; //随机字母

char headline[] = "s w o r d a r t o f f l i n e";

char ETS[] = "PRESS ENTER TO START";

bgm(0); //播放crossing\_field.mp3

settextstyle(30, 8, \_T("SAO UI")); //初始化背景字体和字母颜色

settextcolor(YELLOW);

setlinecolor(BLACK);

for (int i = 0; i <= maxx - 1; i++) /\*\*\*\*\*\*\*\*\*\*\*主循环开始\*\*\*\*\*\*\*\*\*\*\*/

{

hw = GetHWnd();

//计时用变量

tic++;

// 在随机位置显示三个随机字母

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

{

x = (rand() % 134) \* 8;//随机横坐标

y = (rand() % 20) \* 30;//随机纵坐标

c = (rand() % 26) + 65;//随机字母

outtextxy(x, y, c);

}

// 画线擦掉像素行

clearline(i);

//ETS闪烁

if (tic == 50){

blink = !blink;

tic = 0;

}

//获取鼠标位置

GetCursorPos(&pt);

//将获取到的鼠标屏幕位置转换为窗口内位置

ScreenToClient(hw, &pt);

//ETS的鼠标响应判断

if (pt.x >= 411 && pt.y >= 400 && pt.x <= 679 && pt.y <= 450){

flagstart = 1;

if (flagsound == 1){

sound(1);

flagsound = 0;

}

if (KEY\_DOWN(VK\_LBUTTON)){

//sound(0);

bgm(10);

unlimitedMode();

}

}

else{

flagstart = 0;

flagsound = 1;

}

headLine(headline, ETS,blank,flagstart,blink);//输出标题和ETS

Sleep(10); // 延时(控制FPS为100）

if (i >= maxx - 1) i = -1; // 循环处理

if (KEY\_DOWN(VK\_RETURN)){ //ENTER键开始新游戏（暂定）

//sound(0);

bgm(10);

unlimitedMode();

}

if (KEY\_DOWN(VK\_ESCAPE)){ //按ESC键退出

bgm(10);

return true;

closegraph();

}

}/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主循环结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// 关闭图形模式

closegraph();

}

void MAINFRAME::credit(IMAGE \*JY,IMAGE \*HZT,IMAGE \*LTH,int\* num)

{

switch (\*num){

case 1:{

putimage(500, 180, JY);

putimage(300, 180, HZT);

putimage(700, 180, LTH);

}break;

case 2:{

putimage(520, 200, JY);

putimage(320, 200, HZT);

putimage(720, 200, LTH);

}break;

case 3:{

putimage(530, 210, JY);

putimage(330, 210, HZT);

putimage(730, 210, LTH);

}break;

}

}

void MAINFRAME::roadBar(IMAGE \*help)

{

putimage(rightlimit - 510, 200, help);

}

void MAINFRAME::sceneChange(PLAYER \*player,int dir,IMAGE \*background)

{

switch (dir)

{

case 0:{

room--;

if (room < 0){

room = 0;

break;

}

switch (room){

case 0:{

loadimage(background, "pic/composer.jpg");

rightlimit = 1072;

bgm(10);

bgm(8);

}break;

case 1:{

loadimage(background, "pic/blank.jpg");

rightlimit = 3216;

bgm(10);

int num1 = rand() % 2;

switch (num1){

case 0:bgm(2); break;

case 1:bgm(6); break;

}

}break;

case 2:{

loadimage(background, "pic/underground-2.jpg");

rightlimit = 2144;

bgm(10);

int num2 = rand() % 3;

switch (num2){

case 0:bgm(1); break;

case 1:bgm(3); break;

case 2:bgm(7); break;

}

}break;

case 3:{

loadimage(background, "pic/underground-2.jpg");

rightlimit = 2144;

bgm(10);

int num3 = rand() % 3;

switch (num3){

case 0:bgm(1); break;

case 1:bgm(3); break;

case 2:bgm(7); break;

}

}break;

case 4:{

loadimage(background, "pic/underground.jpg");

rightlimit = 2144;

bgm(10);

int num4 = rand() % 3;

switch (num4){

case 0:bgm(1); break;

case 1:bgm(3); break;

case 2:bgm(7); break;

}

}break;

case 5:{

loadimage(background, "pic/boss room.jpg");

rightlimit = 1072;

bgm(10);

int num5 = rand() % 2;

switch (num5){

case 0:bgm(4); break;

case 1:bgm(5); break;

}

}break;

}

originx = -(rightlimit - 1072);

(\*player).teleport(rightlimit - 280, 400);

}break;

case 1:{

room++;

if (room > 5){

room = 5;

break;

}

switch (room){

case 0:{

loadimage(background, "pic/composer.jpg");

rightlimit = 1072;

bgm(10);

bgm(8);

}break;

case 1:{

loadimage(background, "pic/blank.jpg");

rightlimit = 3216;

bgm(10);

int num1 = rand() % 2;

switch (num1){

case 0:bgm(2); break;

case 1:bgm(6); break;

}

}break;

case 2:{

loadimage(background, "pic/underground-2.jpg");

rightlimit = 2144;

bgm(10);

int num2 = rand() % 3;

switch (num2){

case 0:bgm(1); break;

case 1:bgm(3); break;

case 2:bgm(7); break;

}

}break;

case 3:{

loadimage(background, "pic/underground-2.jpg");

rightlimit = 2144;

bgm(10);

int num3 = rand() % 3;

switch (num3){

case 0:bgm(1); break;

case 1:bgm(3); break;

case 2:bgm(7); break;

}

}break;

case 4:{

loadimage(background, "pic/underground.jpg");

rightlimit = 2144;

bgm(10);

int num4 = rand() % 3;

switch (num4){

case 0:bgm(1); break;

case 1:bgm(3); break;

case 2:bgm(7); break;

}

}break;

case 5:{

loadimage(background, "pic/boss room.jpg");

rightlimit = 1072;

bgm(10);

int num5 = rand() % 2;

switch (num5){

case 0:bgm(4); break;

case 1:bgm(5); break;

}

}break;

}

originx = 0;

(\*player).teleport(100, 400);

}break;

case 2:{

loadimage(background, "pic/whiteh.jpg");

room = -1;

rightlimit = 1072;

bgm(10);

bgm(11);

(\*player).teleport(100, 400);

}break;

case 3:{

loadimage(background, "pic/blank.jpg");

room = 1;

rightlimit = 3216;

bgm(10);

int num6 = rand() % 2;

switch (num6){

case 0:bgm(2); break;

case 1:bgm(6); break;

}

(\*player).teleport(1675, 400);

originx = -1200;

}break;

}

}

void MAINFRAME::achievement(PLAYER \*player)

{

settextstyle(60, 16, \_T("SAO UI")); //初始化背景字体和字母颜色

char Floor[30];

char Highfloor[30];

char Mobslaynum[30];

char Deathtime[30];

sprintf\_s(Floor, "FLOOR:%d", floor);

sprintf\_s(Highfloor, "HIGHFLOOR:%d", highfloor);

sprintf\_s(Mobslaynum, "MOB SLAYED NUMBER:%d", mobslaynum);

sprintf\_s(Deathtime, "DEATH:%d", deathtime);

outtextxy(700, 100, Floor);

outtextxy(700, 200, Highfloor);

outtextxy(700, 300, Mobslaynum);

outtextxy(700, 400, Deathtime);

settextstyle(30, 8, \_T("SAO UI")); //初始化背景字体和字母颜色

}

void MAINFRAME::unlimitedMode()

{

IMAGE background, wbackground, welcome, player, skillpic250, skillpic300, enemyplayer, hpUI,soundP,soundHover;

POINT pt; //定义清理图像指针

pt.x = 0; //清理图像指针赋值

pt.y = 0;

//link start!

HWND hwnd = MCIWndCreate(GetHWnd(), NULL, WS\_CHILD | WS\_VISIBLE | MCIWNDF\_NOMENU | MCIWNDF\_NOPLAYBAR, NULL);

MCIWndOpen(hwnd, "avi\\loadgame.wmv", NULL);

MCIWndPlay(hwnd);

Sleep(17000);

settextstyle(30, 8, \_T("SAO UI")); //初始化背景字体和字母颜色

settextcolor(BLACK);

setlinecolor(TRANSPARENT);

setbkmode(TRANSPARENT);

FlushConsoleInputBuffer(GetStdHandle(STD\_INPUT\_HANDLE));

setorigin(originx, 0);//设置初始原点

// 加载通用图片

loadimage(&background, "pic/blank.jpg"); // 请确保该图片是 1072\*600 像素

loadimage(&wbackground, "pic/whitebk.jpg");

loadimage(&soundP, "pic/panel\_sound.jpg");

loadimage(&soundHover, "pic/default\_hover.jpg");

//加载头像

IMAGE HZT, JY, LTH;

loadimage(&HZT, "pic/HZT.jpg");

loadimage(&JY, "pic/JY.jpg");

loadimage(&LTH, "pic/LTH.jpg");

//加载路牌图

IMAGE roadt;

loadimage(&roadt,"pic/roadt.jpg");

// 随机BGM

int m = rand() % 2;

switch (m){

case 0:bgm(2); break;

case 1:bgm(6); break;

}

//计算时间用参数

int tik = 0;

int comboclear = 0;

int ecomboclear = 0;

int mobRefresh = 0;

int AIAttackReady = 0;

int res = 0;

//计算按键触发参数

int flagstart = 0;

int flagsound = 1;

//复活吧，我的勇士

PLAYER kirito;

//为你而战，我的女士

PLAYER enemy;

enemy.setHp(100);

enemy.teleport(0, -200);

enemy.setType(1);

//世界筑造

//开始批量绘图

BeginBatchDraw();

while (1){ /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*游戏主循环\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//获取窗口句柄

hwnd = GetHWnd();

//背景图

putimage(0, 0, &background);

//若处于room=0,制作人员头像抖动

if (room == 0){

tik++;

if (tik > 17)

tik = 0;

if (tik >= 0 && tik <= 11){

int haha = 1;

credit(&HZT, &JY, &LTH, &haha);

}

if (tik >= 12 && tik <= 14){

int haha = 2;

credit(&HZT, &JY, &LTH, &haha);

}

if (tik >= 15 && tik <= 17){

int haha = 3;

credit(&HZT, &JY, &LTH, &haha);

}

}

//路标

if (room == 1 && kirito.getX() <= rightlimit - 455 && kirito.getX() >= rightlimit - 575)

roadBar(&roadt);

//sound开关控制

//获取鼠标位置

GetCursorPos(&pt);

//将获取到的鼠标屏幕位置转换为窗口内位置

ScreenToClient(hwnd, &pt);

//鼠标响应判断

if (pt.x >= 1032 && pt.y >= 10 && pt.x <= 1062 && pt.y <= 40){

flagstart = 1;

if (flagsound == 1){

sound(1);

flagsound = 0;

}

if (KEY\_DOWN(VK\_LBUTTON)){

BGM = false;

SOUND = false;

kirito.setSound(false);

mciSendString(TEXT("stop MySong"), NULL, 0, NULL);

}

if (KEY\_DOWN(VK\_RBUTTON)){

BGM = true;

SOUND = true;

kirito.setSound(true);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}

}

else{

flagstart = 0;

flagsound = 1;

}

if (flagstart==1)

M\_putimg(1032 - originx, 10, &soundHover, WHITE, 80, originx);

M\_putimg(1032 - originx, 10, &soundP, WHITE, 80, originx);

//怪物刷新判定

if (mobRefresh > 0){

mobRefresh--;

if (mobRefresh == 1 && room != 1 && room != 0 && room != -1){

switch (room){

case 2:

case 3:

case 4:{

int newx = (rand() % rightlimit - 180);

enemy.teleport(newx, 400); //随机新位置

int newdir = (rand() % 2);

enemy.setDir(newdir); //随机新方向

int newhp = (rand() % (room \* 30));

enemy.setMaxHp((room \* 300 + newhp)\*floor); //随机新HP

enemy.setHp(enemy.getMaxHp());

enemy.setAttack((room \* 25 + 30)\*floor);

enemy.setLevel(room\*floor);

enemy.setType(9); //新攻击

}break;

case 5:{

int newx = (rand() % rightlimit - 180);

enemy.teleport(newx, 400); //随机新位置

int newdir = (rand() % 2);

enemy.setDir(newdir); //随机新方向

int newhp = (rand() % (room \* 30));

enemy.setMaxHp((room \* 300 + newhp)\*floor); //随机新HP

enemy.setHp(enemy.getMaxHp());

enemy.setAttack((room \* 25 + 30)\*floor);

enemy.setLevel(room\*floor);

enemy.setType(10);

}break;

}

}

}

//摩擦力(Kirito和enemy)

if(kirito.getMovespd()>0&&!kirito.jumpJudge())

kirito.setSpd(kirito.getMovespd() - 1);

if (kirito.getMovespd()<0&&!kirito.jumpJudge())

kirito.setSpd(kirito.getMovespd() + 1);

if (enemy.getMovespd()>0 && !enemy.jumpJudge())

enemy.setSpd(enemy.getMovespd() - 1);

if (enemy.getMovespd()<0 && !enemy.jumpJudge())

enemy.setSpd(enemy.getMovespd() + 1);

//HP被动回复

if (kirito.getHp()<kirito.getMaxHp() && kirito.getHp()>0)

kirito.hpRege();

//HPUI

int khp = kirito.getHp(), kmhp = kirito.getMaxHp();

mainFrame::hpUI(&khp, &kmhp);

int ehp = enemy.getHp(), emhp = enemy.getMaxHp(), ex = enemy.getX(),ey=enemy.getY();

enemyHpUI(&ehp, &emhp, &ex,&ey);

//其他のUI

otherUI(&kirito);

if (room == 1){

char FLOOR[30];

sprintf\_s(FLOOR, "FLOOR:%d", floor);

outtextxy(1758, 290, FLOOR);

}

//成就间

if (room == -1){

achievement(&kirito);

fileRead();

}

//升级判定

kirito.levelUpCheck();

//kirito移动

kirito.moveX(&leftlimit,&rightlimit);

//enemy移动和攻击

int kx = kirito.getX();

if (enemy.getStill() >= -5){

enemy.AIFind(&kx);

if ((AIAttackReady >= 100) && (enemy.stillJudge())){

enemy.AIAttack(&kirito, &enemyplayer);

enemy.setSkillState(10);

enemy.setSkillType(1);

AIAttackReady = 0;

}

}

enemy.moveX(&leftlimit, &rightlimit);

AIAttackReady++;

//检测人物位置，移动屏幕

screenMove(kirito.getX(), kirito.getMovespd());

//静止敌人图片

if (enemy.stillJudge() && !(enemy.getSkillState()) && !enemy.jumpJudge()){

enemy.setRunState(1);

enemy.setSkillType(0);

enemy.setRun(0);

ecomboclear++;

if (ecomboclear == 200){

ecomboclear = 0;

enemy.setCombo(0);

}

int t = enemy.getType();

stillput(enemy.getDir(), enemy.getX(), enemy.getY(), originx, &enemyplayer, t); //静止角色图片

}

//当有键盘输入时执行

if (\_kbhit()){

if (KEY\_DOWN('J') && (!kirito.coolingJudge()) && (!kirito.jumpJudge()) && (!kirito.runJudge())){

int ex = enemy.getX(), eh = enemy.getHp(); //好暴力的取值方法 //普通攻击"J"

kirito.meleeAttack(&ex, &eh, &enemy, &player, &enemyplayer, &skillpic250, &skillpic300, originx);

}

if (KEY\_DOWN('K') && kirito.stillJudge() && (!kirito.coolingJudge())){

int ex = enemy.getX(), eh = enemy.getHp(); //音速冲击"K"

kirito.sonicLeap(&ex, &eh, &enemy, &player, &enemyplayer, &skillpic250, &skillpic300, originx);

}

if (KEY\_DOWN('L') && kirito.stillJudge() && (!kirito.coolingJudge())){

int ex = enemy.getX(), eh = enemy.getHp(); //水平四方斩"L"

kirito.horizontalSquare(&ex, &eh, &enemy, &player, &enemyplayer, &skillpic250, &skillpic300, originx);

}

else if(kirito.getCombo()==0&&kirito.getSkillState()==0&&kirito.getSkillType()==0){

if (KEY\_DOWN(VK\_SPACE) && (!kirito.jumpJudge())){ //按Space跳跃

kirito.startJump();

}

if (KEY\_DOWN('A')){ //按A向左移动

kirito.setDir(0); //改变方向

kirito.setStill(-5); //开始跑动

kirito.setRunState(1); //使runState为1

kirito.setSpd(-10); //使kirito获得速度

if (KEY\_DOWN(VK\_SHIFT)) //配合shift食用更佳

kirito.setSpd(-15);

}

if (KEY\_DOWN('D')){ //按D向右移动

kirito.setDir(1); //改变方向

kirito.setStill(-5); //开始跑动

kirito.setRunState(1); //使runState为1

kirito.setSpd(10);

if (KEY\_DOWN(VK\_SHIFT))

kirito.setSpd(15);

}

if (KEY\_DOWN('W')){ //场景转换

if (kirito.getX() - leftlimit <= 50){

sceneChange(&kirito, 0, &background);

enemy.setHp(-1);

mobRefresh = 100;

enemy.teleport(0, -200);

}

if (rightlimit - kirito.getX() <= 230){

sceneChange(&kirito, 1, &background);

enemy.setHp(-1);

mobRefresh = 100;

enemy.teleport(0, -200);

}

if (kirito.getX() <= 1750 && kirito.getX()>=1600&&room==1){

sceneChange(&kirito, 2, &background);

enemy.setHp(-1);

mobRefresh = 100;

enemy.teleport(0, -200);

}

if (kirito.getX() <= 550 && kirito.getX() >= 380&&room == -1){

sceneChange(&kirito, 3, &background);

enemy.setHp(-1);

mobRefresh = 100;

enemy.teleport(0, -200);

}

}

if (KEY\_DOWN(VK\_ESCAPE)){ //ESC退出

bgm(10);

break;

}

if (KEY\_DOWN('R')){

kirito.respawn();

}

}

}

//若kirito处于跳跃状态，进行Y值改变和绘图

if (kirito.jumpJudge()){

kirito.jump();

switch (kirito.getDir()){

case 0:loadimage(&player, "pic/ll4.jpg",180,185,true); break;

case 1:loadimage(&player, "pic/rr4.jpg", 180, 185, true); break;

}

kirito.setCombo(0);

kirito.setSkillType(0);

kirito.setSkillState(0);

M\_putimg(kirito.getX(), kirito.getY(), &player, WHITE, 100, originx);

}

//若kirito处于跑动状态，进行跑步绘图

if (kirito.runJudge() && !kirito.jumpJudge() && !kirito.stillJudge()){

kirito.running(&player, originx,&kirito);

}

if (enemy.runJudge() && !enemy.jumpJudge() && (enemy.getStill()>=-5)){

enemy.running(&enemyplayer, originx,&enemy);

}

//技能释放时的姿势和特效绘制

kirito.skillEffect(&skillpic250,&skillpic300,&player,&enemyplayer,&enemy,originx);

//between

if (!kirito.stillJudge()){

kirito.setStill(kirito.getStill() + 1);

M\_putimg(kirito.getX(), kirito.getY(), &player, WHITE, 100, originx);

}

//enemy between

if (!enemy.stillJudge()){

enemy.setStill(enemy.getStill() + 1);

if (enemy.getType()==10)

switch (enemy.getCombo()){

case 1:M\_putimg(enemy.getX(), enemy.getY() - 425 + 185, &enemyplayer, WHITE, 100, originx); break;

case 2:M\_putimg(enemy.getX(), enemy.getY() - 425 + 185, &enemyplayer, WHITE, 100, originx); break;

case 3:M\_putimg(enemy.getX(), enemy.getY() - 510 + 185, &enemyplayer, WHITE, 100, originx); break;

case 4:M\_putimg(enemy.getX(), enemy.getY() - 375 + 185, &enemyplayer, WHITE, 100, originx); break;

}

else

M\_putimg(enemy.getX(), enemy.getY(), &enemyplayer, WHITE, 100, originx);

}

//静止时的putimg

if (kirito.stillJudge() && !(kirito.getSkillState()) && !kirito.jumpJudge()){

kirito.setRunState(1);

kirito.setSkillType(0);

kirito.setRun(0);

comboclear++;

if (comboclear == 20){

comboclear = 0;

kirito.setCombo(0);

}

stillput(kirito.getDir(),kirito.getX(),kirito.getY(),originx, &player,0); //静止角色图片

}

//死亡判定

if (kirito.getHp() <= 0){

kirito.setHp(0);

kirito.teleport(kirito.getX(), -200);

if (res==0)

deathtime++;

res++;

if (res == 500){

while (room > 1)

sceneChange(&kirito, 0, &background);

enemy.setHp(-1);

mobRefresh = 100;

enemy.teleport(0, -200);

kirito.teleport(300, 400);

kirito.respawn();

originx = 0;

}

}

if ((enemy.getHp() <= 0) && (mobRefresh == 0) && room != 1 && room != 0 && room != -1){

enemy.teleport(0, -200);

mobRefresh = 100;

switch (enemy.getType()){

case 1: //普通怪死亡

case 2:

case 3:

case 4:

case 5:

case 6:

case 7:

case 8:

case 9:{

kirito.expAdd(10 \* enemy.getLev());

mobslaynum++;

}break;

case 10:{ //BOSS死亡

kirito.expAdd(100 \* enemy.getLev());

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

sceneChange(&kirito, 0, &background);

}

enemy.setHp(-1);

enemy.teleport(0, -200);

floor++;

mobslaynum++;

if (highfloor<floor)highfloor = floor;

}break;

}

}

//冷却时间减少

if (kirito.coolingJudge())

kirito.coolingDown();

if (kirito.getSkillState() > 0)

kirito.skillStateMove();

if (enemy.coolingJudge())

enemy.coolingDown();

if (enemy.getSkillState() > 0)

enemy.skillStateMove();

FlushBatchDraw(); //绘制结果输出

Sleep(inter); //控制帧率

cleardevice(); //清空画面

}//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*主循环结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

closegraph();

//welcomeInit();

}

void MAINFRAME::bgm(int song)

{

if (BGM)

switch (song){

case 0:{

mciSendString(TEXT("open bgm/crossing\_field.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 1:{

mciSendString(TEXT("open bgm/Swordland.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 2:{

mciSendString(TEXT("open bgm/The\_First\_Town.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 3:{

mciSendString(TEXT("open bgm/Survive\_The\_Swordland.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 4:{

mciSendString(TEXT("open bgm/Luminous\_Sword.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 5:{

mciSendString(TEXT("open bgm/light\_your\_sword.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 6:{

mciSendString(TEXT("open bgm/Everyday\_Life.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 7:{

mciSendString(TEXT("open bgm/She\_has\_to\_overcome\_her\_fear.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 8:{

mciSendString(TEXT("open bgm/X.U..mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 11:{

mciSendString(TEXT("open bgm/Memory\_Heart\_Message.mp3 alias MySong"), NULL, 0, NULL);

mciSendString(TEXT("play MySong"), NULL, 0, NULL);

}break;

case 10:{

mciSendString(TEXT("close MySong"), NULL, 0, NULL);

}

case 9:{

mciSendString(TEXT("stop MySong"), NULL, 0, NULL);

}

default:break;

}

else

switch (song){

case 10:{

mciSendString(TEXT("close MySong"), NULL, 0, NULL);

}

case 9:{

mciSendString(TEXT("stop MySong"), NULL, 0, NULL);

}

default:break;

}

}

void MAINFRAME::sound(int soundtype)

{

if (SOUND)

switch (soundtype){

case 0:{

PlaySound("sound/boot.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 1:{

PlaySound("sound/press.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 2:{

PlaySound("sound/melee\_1.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 3:{

PlaySound("sound/welcome.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 4:{

PlaySound("sound/tele.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 5:{

PlaySound("sound/jump.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 6:{

PlaySound("sound/four.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 7:{

PlaySound("sound/res.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 8:{

PlaySound("sound/xiu.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

default:break;

}

}

void MAINFRAME::gameExit()

{

bgm(9);

closegraph();

}

void MAINFRAME::hpUI(int \*hp, int \*maxHp)

{

IMAGE hpTIAO, hpCAO;

float hpRatio = (1.0\*(\*hp)) / (1.0\*(\*maxHp));

int hpPix = hpRatio \* 258;

char hps[30] = "\0";

if (hpRatio > 0.66) //根据血量加载适应颜色的hpTIAO

loadimage(&hpTIAO, "pic/hp\_green.jpg",hpPix,24,true);

if (hpRatio > 0.33&&hpRatio <= 0.66)

loadimage(&hpTIAO, "pic/hp\_yellow.jpg", hpPix, 24, true);

if (hpRatio <= 0.33)

loadimage(&hpTIAO, "pic/hp\_red.jpg", hpPix, 24, true);

loadimage(&hpCAO, "pic/hp\_bar.jpg"); //加载hpCAO

sprintf\_s(hps, "%d / %d",\*hp,\*maxHp);

//图像输出

if (hp > 0){

M\_putimg(75 - originx, 12, &hpTIAO, WHITE, 100, originx);

HPCAO\_putimg(0 - originx, 0, &hpCAO, WHITE, 80, originx);

outtextxy(250-originx,33,hps);

}

}

void MAINFRAME::otherUI(PLAYER\* player)

{

char movespd[20];

char still[20];

char combo[10];

char skilltype[20];

char skillstate[20];

char runstate[20];

char level[30];

char exp[30];

char atk[30];

char x[20];

sprintf\_s(movespd, "MOVESPD:%d", (\*player).getMovespd());

sprintf\_s(still, "STILL:%d", (\*player).getStill());

sprintf\_s(combo, "COMBO:%d", (\*player).getCombo());

sprintf\_s(skilltype, "SKILLTYPE:%d", (\*player).getSkillType());

sprintf\_s(skillstate, "SKILLSTATE:%d", (\*player).getSkillState());

sprintf\_s(runstate, "RUNSTATE:%d", (\*player).getRunState());

sprintf\_s(level, "LEVEL:%d", (\*player).getLev());

sprintf\_s(exp, "EXP:%d / %d", (\*player).getExp(), 25\*(\*player).getLev()\*(\*player).getLev() + (\*player).getLev() \* 5 + 20);

sprintf\_s(atk, "ATTACK:%d", (\*player).getAttack());

sprintf\_s(x, "X:%d", (\*player).getX());

outtextxy(-originx, 50, movespd);

outtextxy(-originx, 90, still);

outtextxy(-originx, 130, combo);

outtextxy(-originx, 170, skilltype);

outtextxy(-originx, 210, skillstate);

outtextxy(-originx, 370, runstate);

outtextxy(-originx, 250, level);

outtextxy(-originx, 290, exp);

outtextxy(-originx, 330, atk);

outtextxy(-originx, 410, x);

}

void MAINFRAME::enemyHpUI(int \*hp, int \*maxHp,int \*x,int \*y)

{

IMAGE hpTIAO;

float hpRatio = (1.0\*(\*hp)) / (1.0\*(\*maxHp));

int hpPix = hpRatio \* 180;

char hpNum[20];

sprintf\_s(hpNum, "%d / %d", \*hp,\*maxHp);

if (hpRatio > 0.66) //根据血量加载适应颜色的hpTIAO

loadimage(&hpTIAO, "pic/hp\_enemy\_green.jpg", hpPix, 24, true);

if (hpRatio > 0.33&&hpRatio <= 0.66)

loadimage(&hpTIAO, "pic/hp\_enemy\_yellow.jpg", hpPix, 24, true);

if (hpRatio <= 0.33)

loadimage(&hpTIAO, "pic/hp\_enemy\_red.jpg", hpPix, 24, true);

//图像输出

M\_putimg(\*x, \*y-20, &hpTIAO, WHITE, 100, originx);

outtextxy(\*x+180, \*y - 20, hpNum);

}

void MAINFRAME::bossHpUI(int \*hp, int \*maxHp, int \*x, int \*y)

{

IMAGE hpTIAO;

float hpRatio = (1.0\*(\*hp)) / (1.0\*(\*maxHp));

int hpPix = hpRatio \* 180;

char hpNum[20];

sprintf\_s(hpNum, "%d / %d", \*hp, \*maxHp);

if (hpRatio > 0.66) //根据血量加载适应颜色的hpTIAO

loadimage(&hpTIAO, "pic/hp\_enemy\_green.jpg", hpPix, 24, true);

if (hpRatio > 0.33&&hpRatio <= 0.66)

loadimage(&hpTIAO, "pic/hp\_enemy\_yellow.jpg", hpPix, 24, true);

if (hpRatio <= 0.33)

loadimage(&hpTIAO, "pic/hp\_enemy\_red.jpg", hpPix, 24, true);

//图像输出

M\_putimg(\*x, \*y - 260, &hpTIAO, WHITE, 100, originx);

outtextxy(\*x + 180, \*y - 260, hpNum);

}

void MAINFRAME::M\_putimg(int dstX, int dstY, IMAGE \*pimg, int avoid\_color, int tp, int originx)

{

//排除颜色avoid\_color,容差为deviation；透明度tp(transparency)从0到100

setorigin(originx, 0);

int x, y, num;

int deviation = 100;

int R, G, B;//记录贴图某点色彩

//记录排除颜色色彩

int avoid\_r = GetRValue(avoid\_color);

int avoid\_g = GetGValue(avoid\_color);

int avoid\_b = GetBValue(avoid\_color);

IMAGE pSrcImg;//背景图

IMAGE tempimg;//临时贴图

mainFrame::copy\_img(&tempimg, pimg);//保护原图

SetWorkingImage(NULL);//对默认绘图窗口的绘图操作

getimage(&pSrcImg, dstX, dstY, pimg->getwidth(), pimg->getheight());

//透明度容错

if (tp<0)

{

tp = 0;

}

else if (tp>100)

{

tp = 100;

}

// 获取背景指向显存的指针

DWORD\* bk\_pMem = GetImageBuffer(&pSrcImg);

//贴图指向显存的指针

DWORD\* pMem = GetImageBuffer(&tempimg);

for (y = 0; y<pimg->getheight(); y++)

{

for (x = 0; x<pimg->getwidth(); x++)

{

num = y\*pimg->getwidth() + x;

R = GetRValue(pMem[num]);

G = GetGValue(pMem[num]);

B = GetBValue(pMem[num]);

if ((abs(R - avoid\_r) <= deviation) && (abs(G - avoid\_g) <= deviation) && (abs(B - avoid\_b) <= deviation))

{

pMem[num] = bk\_pMem[num];

}

else

{

pMem[num] = RGB((R\*tp + GetRValue(bk\_pMem[num])\*(100 - tp)) / 100, (G\*tp + GetGValue(bk\_pMem[num])\*(100 - tp)) / 100, (B\*tp + GetBValue(bk\_pMem[num])\*(100 - tp)) / 100);

}

}

}

putimage(dstX, dstY, &tempimg);

}

void MAINFRAME::HPCAO\_putimg(int dstX, int dstY, IMAGE \*pimg, int avoid\_color, int tp, int originx)

{

//排除颜色avoid\_color,容差为deviation；透明度tp(transparency)从0到100

setorigin(originx, 0);

int x, y, num;

int deviation = 10;

int R, G, B;//记录贴图某点色彩

//记录排除颜色色彩

int avoid\_r = GetRValue(avoid\_color);

int avoid\_g = GetGValue(avoid\_color);

int avoid\_b = GetBValue(avoid\_color);

IMAGE pSrcImg;//背景图

IMAGE tempimg;//临时贴图

mainFrame::copy\_img(&tempimg, pimg);//保护原图

SetWorkingImage(NULL);//对默认绘图窗口的绘图操作

getimage(&pSrcImg, dstX, dstY, pimg->getwidth(), pimg->getheight());

//透明度容错

if (tp<0)

{

tp = 0;

}

else if (tp>100)

{

tp = 100;

}

// 获取背景指向显存的指针

DWORD\* bk\_pMem = GetImageBuffer(&pSrcImg);

//贴图指向显存的指针

DWORD\* pMem = GetImageBuffer(&tempimg);

for (y = 0; y<pimg->getheight(); y++)

{

for (x = 0; x<pimg->getwidth(); x++)

{

num = y\*pimg->getwidth() + x;

R = GetRValue(pMem[num]);

G = GetGValue(pMem[num]);

B = GetBValue(pMem[num]);

if ((abs(R - avoid\_r) <= deviation) && (abs(G - avoid\_g) <= deviation) && (abs(B - avoid\_b) <= deviation))

{

pMem[num] = bk\_pMem[num];

}

else

{

pMem[num] = RGB((R\*tp + GetRValue(bk\_pMem[num])\*(100 - tp)) / 100, (G\*tp + GetGValue(bk\_pMem[num])\*(100 - tp)) / 100, (B\*tp + GetBValue(bk\_pMem[num])\*(100 - tp)) / 100);

}

}

}

putimage(dstX, dstY, &tempimg);

}

Player.cpp

#include "player.h"

#include "mainframe.h"

#include <math.h>

#define KEY\_DOWN(vk\_c) (GetAsyncKeyState(vk\_c)&0x8000?1:0)

#define INIT\_X 300

#define INIT\_Y 400

#define INIT\_DIR 1

#define INIT\_MAXHP 250

#define INIT\_HP 250

#define INIT\_ATTACK 50

#define INIT\_MOVESPD 0

#define INIT\_SKILLSTATE 0

#define INIT\_SKILLTYPE 0

#define INIT\_ISRUN false

#define INIT\_ISJUMP false

#define INIT\_COOLDOWN 0

#define INIT\_RUNSTATE 0

#define INIT\_JUMPSTATE 0

#define INIT\_STILL 1

#define INIT\_RUN 0

#define INIT\_COMBO 0

#define INIT\_EXP 0

#define INIT\_LEVEL 1

#define INIT\_HPRE 0.1

#define INIT\_SONICLEAP\_RANGE 800

#define INIT\_MELEEATTACK\_RANGE 150

#define INIT\_HS\_RANGE 200

#define INIT\_FINDRANGE 500

PLAYER::PLAYER()

{

x = INIT\_X; //初始化\_\_PLAYER数据

y = INIT\_Y;

dir = INIT\_DIR;

maxhp = INIT\_MAXHP;

hp = maxhp;

attack = INIT\_ATTACK;

movespd = INIT\_MOVESPD;

skillState = INIT\_SKILLSTATE;

skillType = INIT\_SKILLTYPE;

isRun = INIT\_ISRUN;

isJump = INIT\_ISJUMP;

coolDown = INIT\_COOLDOWN;

runState = INIT\_RUNSTATE;

jumpState = INIT\_JUMPSTATE;

still = INIT\_STILL;

run = INIT\_RUN;

combo = INIT\_COMBO;

exp = INIT\_EXP;

level = INIT\_LEVEL;

hpregen = INIT\_HPRE;

type = 0;

}

PLAYER::~PLAYER() {}

int PLAYER::getX() //获取位置

{

return x;

}

int PLAYER::getY()

{

return y;

}

int PLAYER::getMovespd()//获取移动速度

{

return movespd;

}

int PLAYER::getAttack() //获取攻击力

{

return attack;

}

float PLAYER::getHp() //获取当前hp

{

return hp;

}

int PLAYER::getCombo() //获取当前连击值

{

return combo;

}

int PLAYER::getSkill()

{

return skillType;

}

int PLAYER::getRun()

{

return run;

}

void PLAYER::jump() //跳跃

{

y = (int)((15 - jumpState)\*(15 - jumpState)\*0.6 + 265);

jumpState++;

if (jumpState > 30){

jumpState = 0;

isJump = false;

}

}

void PLAYER::running(IMAGE\* player, int ox,PLAYER \*p)

{

switch ((\*p).getType()){

case 0:{

switch ((\*p).getDir()){

case 1:{

(\*p).setRun((\*p).getRun() + 1);

if ((\*p).getRun() >= 17)(\*p).setRun(1);

if ((\*p).getRun() >= 4)(\*p).setRunState(2);

if ((\*p).getRun() >= 8)(\*p).setRunState(3);

if ((\*p).getRun() >= 12)(\*p).setRunState(4);

if ((\*p).getRun() >= 16)(\*p).setRunState(1);

switch ((\*p).getRunState()){

case 1:loadimage(player, "pic/rr1.jpg", 180, 185, true); break;

case 2:loadimage(player, "pic/rr2.jpg", 180, 185, true); break;

case 3:loadimage(player, "pic/rr3.jpg", 180, 185, true); break;

case 4:loadimage(player, "pic/rr4.jpg", 180, 185, true); break;

}

MAINFRAME::M\_putimg((\*p).getX(), (\*p).getY(), player, WHITE, 100, ox);

}break;

case 0:{

(\*p).setRun((\*p).getRun() + 1);

if ((\*p).getRun() >= 17)(\*p).setRun(1);

if ((\*p).getRun() >= 4)(\*p).setRunState(2);

if ((\*p).getRun() >= 8)(\*p).setRunState(3);

if ((\*p).getRun() >= 12)(\*p).setRunState(4);

if ((\*p).getRun() >= 16)(\*p).setRunState(1);

switch ((\*p).getRunState()){

case 1:loadimage(player, "pic/ll1.jpg", 180, 185, true); break;

case 2:loadimage(player, "pic/ll2.jpg", 180, 185, true); break;

case 3:loadimage(player, "pic/ll3.jpg", 180, 185, true); break;

case 4:loadimage(player, "pic/ll4.jpg", 180, 185, true); break;

}

MAINFRAME::M\_putimg((\*p).getX(), (\*p).getY(), player, WHITE, 100, ox);

}break;

}

}break;

case 9:{

switch ((\*p).getDir()){

case 1:{

(\*p).setRun((\*p).getRun() + 1);

if ((\*p).getRun() >= 17)(\*p).setRun(1);

if ((\*p).getRun() >= 4)(\*p).setRunState(2);

if ((\*p).getRun() >= 8)(\*p).setRunState(3);

if ((\*p).getRun() >= 12)(\*p).setRunState(4);

if ((\*p).getRun() >= 16)(\*p).setRunState(1);

switch ((\*p).getRunState()){

case 1:loadimage(player, "pic/enemyrrunning1.jpg", 180, 185, true); break;

case 2:loadimage(player, "pic/enemyrrunning2.jpg", 180, 185, true); break;

case 3:loadimage(player, "pic/enemyrrunning3.jpg", 180, 185, true); break;

case 4:loadimage(player, "pic/enemyrrunning4.jpg", 180, 185, true); break;

}

MAINFRAME::M\_putimg((\*p).getX(), (\*p).getY(), player, WHITE, 100, ox);

}break;

case 0:{

(\*p).setRun((\*p).getRun() + 1);

if ((\*p).getRun() >= 17)(\*p).setRun(1);

if ((\*p).getRun() >= 4)(\*p).setRunState(2);

if ((\*p).getRun() >= 8)(\*p).setRunState(3);

if ((\*p).getRun() >= 12)(\*p).setRunState(4);

if ((\*p).getRun() >= 16)(\*p).setRunState(1);

switch ((\*p).getRunState()){

case 1:loadimage(player, "pic/enemylrunning1.jpg", 180, 185, true); break;

case 2:loadimage(player, "pic/enemylrunning2.jpg", 180, 185, true); break;

case 3:loadimage(player, "pic/enemylrunning3.jpg", 180, 185, true); break;

case 4:loadimage(player, "pic/enemylrunning4.jpg", 180, 185, true); break;

}

MAINFRAME::M\_putimg((\*p).getX(), (\*p).getY(), player, WHITE, 100, ox);

}break;

}

}break;

}

}

void PLAYER::moveX(int \*leftlimit,int \*rightlimit) //基本移动

{

x += movespd;

switch (dir){

case 1:{

if (x > (\*rightlimit - 190)){

x = (\*rightlimit - 190);

}

}break;

case 0:{

if (x <(10 + \*leftlimit)){

x = (10 + \*leftlimit);

}

}break;

}

}

bool PLAYER::stillJudge() //判断是否静止

{

if (still==1)

return true;

else if (still<1)

return false;

}

bool PLAYER::runJudge()

{

if(isRun)

return true;

return false;

}

bool PLAYER::jumpJudge()

{

if (isJump)

return true;

else

return false;

}

bool PLAYER::getDir()

{

return dir;

}

void PLAYER::startJump()

{

isJump = true;

jumpState = 1;

soundEffect(5);

}

void PLAYER::useSkill(int skillnum)

{

switch (skillnum){

case 1:skillType = 1, skillState = 1; break;

default:break;

}

}

void PLAYER::meleeAttack(int \*enemyx, int \*enemyhp, PLAYER \*enemy, IMAGE \*player, IMAGE \*enemyplayer, IMAGE \*skillpic250, IMAGE \*skillpic300, int originx)

{

skillType = 1;

skillState = 10;

coolDown = 10;

//攻击判定

combo++;

still = -10;

soundEffect(2);

switch (dir){

case 0:{

if ((x - (\*enemy).getX()) <= INIT\_MELEEATTACK\_RANGE && (x - (\*enemy).getX()) >= 0){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - attack); //击中造成伤害

(\*enemy).setStill(-15); //击中造成僵直

(\*enemy).setSpd(-10);

(\*enemy).setRun(0); //击中造成击退

if ((\*enemy).getType()==9){

loadimage(enemyplayer, "pic/enemyrhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, originx); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

switch (combo){

case 1:{

loadimage(player, "pic/lattack1.jpg");

loadimage(skillpic250, "pic/lcut1.jpg");

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

case 2:{

loadimage(player, "pic/lattack2.jpg");

loadimage(skillpic250, "pic/lcut2.jpg");

x -= 50;

movespd = -10;

if (x < 10)x = 10;

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

case 3:{

loadimage(player, "pic/lattack3.jpg");

loadimage(skillpic300, "pic/lcut3.jpg");

movespd = -4 ;

if (x < 10)x = 10;

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

}

}break;

case 1:{

if (((\*enemy).getX() - x) <= INIT\_MELEEATTACK\_RANGE && ((\*enemy).getX() - x) >= 0){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - attack); //击中造成伤害

(\*enemy).setStill(-15); //击中造成僵直

(\*enemy).setSpd(10);

(\*enemy).setRun(0); //击中造成击退

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemylhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, originx); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

switch (combo){

case 1:{

loadimage(player, "pic/rattack1.jpg");

loadimage(skillpic250, "pic/rcut1.jpg");

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

case 2:{

loadimage(player, "pic/rattack2.jpg");

loadimage(skillpic250, "pic/rcut2.jpg");

x += 50;

movespd = 10;

if (x < 10)x = 10;

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

case 3:{

loadimage(player, "pic/rattack3.jpg");

loadimage(skillpic300, "pic/rcut3.jpg");

movespd = 4;

if (x < 10)x = 10;

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

}

}break;

}

}

void PLAYER::sonicLeap(int \*enemyx, int \*enemyhp, PLAYER \*enemy, IMAGE \*player, IMAGE \*enemyplayer, IMAGE \*skillpic250, IMAGE \*skillpic300, int originx)

{

skillType = 2;

skillState = 50;

coolDown = 50;

//攻击判定

combo++;

soundEffect(8);

still = -50;

switch (dir){

case 0:{

switch (combo){

case 1:{

loadimage(player, "pic/lsonicleap-2.jpg");

loadimage(skillpic250, "pic/lsonicleap-1.jpg");

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

}

}break;

case 1:{

switch (combo){

case 1:{

loadimage(player, "pic/rsonicleap-2.jpg");

loadimage(skillpic250, "pic/rsonicleap-1.jpg");

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

}

}break;

}

}

void PLAYER::horizontalSquare(int \*enemyx, int \*enemyhp, PLAYER \*enemy, IMAGE \*player, IMAGE \*enemyplayer, IMAGE \*skillpic250, IMAGE \*skillpic300, int originx)

{

skillType = 3;

skillState = 50;

coolDown = 50;

//攻击判定

combo++;

still = -50;

soundEffect(2);

switch (dir){

case 0:{

switch (combo){

case 1:{

loadimage(player, "pic/fourdash-2.jpg");

loadimage(skillpic250, "pic/fourdash-1.jpg");

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

}

}break;

case 1:{

switch (combo){

case 1:{

loadimage(player, "pic/rfourdash-2.jpg");

loadimage(skillpic250, "pic/rfourdash-1.jpg");

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, originx);

}break;

}

}break;

}

}

int PLAYER::getStill()

{

return still;

}

void PLAYER::restill()

{

if (still<1)

still++;

if (still == 1){

isRun = false;

run = 0;

runState = 0;

}

}

int PLAYER::getSkillState()

{

return skillState;

}

int PLAYER::getRunState()

{

return runState;

}

void PLAYER::skillEffect(IMAGE \*skillpic250, IMAGE \*skillpic300, IMAGE \*player, IMAGE \*enemyplayer, PLAYER \*enemy, int ox)

{

switch (skillType){ //判断技能种类(0为无技能)

case 1:{ //普通攻击

switch (dir){

case 0:{

switch (combo){ //根据combo判断技能图片

case 1:{

y -= 25; //技能图片位置修正

x -= 85;

loadimage(skillpic250, "pic/lcut1.jpg", 250, 250, true);

MAINFRAME::M\_putimg(x, y, skillpic250, WHITE, 10 \* skillState, ox);

y += 25; //技能图片位置修正后还原

x += 85;

}break;

case 2:{

y -= 50;

x -= 65;

loadimage(skillpic250, "pic/lcut2.jpg", 250, 250, true);

MAINFRAME::M\_putimg(x, y, skillpic250, WHITE, 10 \* skillState, ox);

y += 50;

x += 65;

}break;

case 3:{

y -= 80;

x -= 105;

loadimage(skillpic300, "pic/lcut3.jpg", 300, 300, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 10 \* skillState, ox);

y += 80;

x += 105;

if (combo == 3 && skillState == 1)combo = 0;

}break;

}

}break;

case 1:{

switch (combo){ //根据combo判断技能图片

case 1:{

y -= 25; //技能图片位置修正

x += 0;

loadimage(skillpic250, "pic/rcut1.jpg", 250, 250, true);

MAINFRAME::M\_putimg(x, y, skillpic250, WHITE, 10 \* skillState, ox);

y += 25; //技能图片位置修正后还原

x -= 0;

}break;

case 2:{

y -= 47;

x += 0;

loadimage(skillpic250, "pic/rcut2.jpg", 250, 250, true);

MAINFRAME::M\_putimg(x, y, skillpic250, WHITE, 10 \* skillState, ox);

y += 47;

x -= 0;

}break;

case 3:{

y -= 82;

x -= 0;

loadimage(skillpic300, "pic/rcut3.jpg", 300, 300, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 10 \* skillState, ox);

y += 82;

x += 0;

if (combo == 3 && skillState == 1)combo = 0;

}break;

}

}break;

}

}break;//技能1结束

case 2:{ //音速冲击

switch (dir){

case 0:{

if (skillState <= 30)combo = 2; //由skillState确定combo

if (skillState <= 20)combo = 3;

switch (combo){ //根据combo判断技能图片

case 1:{

//技能图片位置修正

loadimage(skillpic250, "pic/lsonicleap-1.jpg", 180, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic250, WHITE, 5 \* (skillState - 30), ox);

//技能图片位置修正后还原

}break;

case 2:{

if (skillState == 30){

x -= 110;

movespd = -80;

}

loadimage(player, "pic/lsonicleap-3.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

x -= 340;

loadimage(skillpic300, "pic/sonicleap\_flash.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 10 \* (skillState - 20), ox);

if ((x - (\*enemy).getX()) <= INIT\_SONICLEAP\_RANGE && (x - (\*enemy).getX()) >= 0){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-50); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemyrhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

x += 340;

}break;

case 3:{

if (skillState == 20)x -= 110;

loadimage(player, "pic/lsonicleap-4.jpg", 354, 185, true);

movespd = -2;

if (skillState == 1){

combo = 0;

x += 110;

}

if (skillState == 1 || skillState == 0)

loadimage(player, "pic/lstill.jpg", 180, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

}break;

}

}break;

case 1:{

if (skillState <= 30)combo = 2; //由skillState确定combo

if (skillState <= 20)combo = 3;

switch (combo){ //根据combo判断技能图片

case 1:{

//技能图片位置修正

loadimage(skillpic250, "pic/rsonicleap-1.jpg", 180, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic250, WHITE, 5 \* (skillState - 30), ox);

//技能图片位置修正后还原

}break;

case 2:{

if (skillState == 30){

x -= 110;

movespd = 80;

}

loadimage(player, "pic/rsonicleap-3.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

loadimage(skillpic300, "pic/sonicleap\_flash.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 10 \* (skillState - 20), ox);

if (((\*enemy).getX() - x) <= INIT\_SONICLEAP\_RANGE && ((\*enemy).getX() - x) >= 0){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-50); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemylhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

}break;

case 3:{

if (skillState == 20)x -= 110;

loadimage(player, "pic/rsonicleap-4.jpg", 354, 185, true);

movespd = 2;

if (skillState == 1){

combo = 0;

x += 110;

}

if (skillState == 1 || skillState == 0)

loadimage(player, "pic/rstill.jpg", 180, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

}break;

}

}break;

}

}break;//技能2结束

case 3:{ //水平四方斩horizontalsquare

switch (dir){

case 0:{

if (skillState <= 35)combo = 2; //由skillState确定combo

if (skillState <= 30)combo = 3;

if (skillState <= 25)combo = 4;

if (skillState <= 20)combo = 5;

if (skillState <= 15)combo = 6;

switch (combo){ //根据combo判断技能图片

case 1:{

//技能图片位置修正

loadimage(skillpic250, "pic/fourdash-1.jpg", 180, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic250, WHITE, 6 \* (skillState - 35), ox);

Sleep(30);

//技能图片位置修正后还原

}break;

case 2:{ //第1击

if (skillState == 35){

x -= 110;

movespd = -150;

soundEffect(6);

}

if (skillState < 34){

movespd = 0;

Sleep(10);

}

if ((x - (\*enemy).getX()) <= INIT\_HS\_RANGE || ((\*enemy).getX() - x) <= INIT\_HS\_RANGE){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-90); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemyrhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

loadimage(player, "pic/fourdash-3.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

x -= 75; //加特技

loadimage(skillpic300, "pic/foursp-1.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 20 \* (skillState - 30), ox);

x += 75;

}break;

case 3:{ //第2击

if (skillState == 30){

movespd = 150;

}

if (skillState < 29){

movespd = 0;

Sleep(30);

}

if ((x - (\*enemy).getX()) <= INIT\_HS\_RANGE || ((\*enemy).getX() - x) <= INIT\_HS\_RANGE){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-90); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemylhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

loadimage(player, "pic/fourdash-4.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

x -= 25;

y -= 20;

loadimage(skillpic300, "pic/foursp-2.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 20 \* (skillState - 25), ox);

x += 25;

y += 20;

}break;

case 4:{ //第3击

if (skillState == 25){

movespd = -150;

}

if (skillState < 24){

movespd = 0;

Sleep(30);

}

if ((x - (\*enemy).getX()) <= INIT\_HS\_RANGE || ((\*enemy).getX() - x) <= INIT\_HS\_RANGE){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-90); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemyrhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

loadimage(player, "pic/fourdash-5.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

y -= 30;

x -= 13;

loadimage(skillpic300, "pic/foursp-3.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 20 \* (skillState - 20), ox);

y += 30;

x += 13;

}break;

case 5:{ //第4击

if (skillState == 20){

movespd = 150;

}

if (skillState < 19){

movespd = 0;

Sleep(30);

}

if ((x - (\*enemy).getX()) <= INIT\_HS\_RANGE || ((\*enemy).getX() - x) <= INIT\_HS\_RANGE){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-90); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemylhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

loadimage(player, "pic/fourdash-6.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

loadimage(skillpic300, "pic/foursp-4.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 20 \* (skillState - 15), ox);

}break;

case 6:{ //收尾

loadimage(player, "pic/fourdash-6.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

if (skillState > 0 && (skillState < 10))

loadimage(skillpic300, "pic/foursp-5-6.jpg", 340, 185, true);

if (skillState>3 && (skillState < 10))

loadimage(skillpic300, "pic/foursp-5-4.jpg", 340, 185, true);

if ((skillState>6) && (skillState < 10))

loadimage(skillpic300, "pic/foursp-5.jpg", 340, 185, true);

MAINFRAME::M\_putimg((\*enemy).getX() - 110, y, skillpic300, WHITE, 6 \* skillState, ox);

if (skillState == 1){

combo = 0;

x += 110;

}

if (skillState == 1 || skillState == 0){

loadimage(player, "pic/rstill.jpg", 180, 185, true);

dir = 1;

}

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

}break;

}

}break;

case 1:{

if (skillState <= 35)combo = 2; //由skillState确定combo

if (skillState <= 30)combo = 3;

if (skillState <= 25)combo = 4;

if (skillState <= 20)combo = 5;

if (skillState <= 15)combo = 6;

switch (combo){ //根据combo判断技能图片

case 1:{

//技能图片位置修正

loadimage(skillpic250, "pic/rfourdash-1.jpg", 180, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic250, WHITE, 6 \* (skillState - 35), ox);

Sleep(30);

//技能图片位置修正后还原

}break;

case 2:{ //第1击

if (skillState == 35){

x -= 110;

movespd = +150;

soundEffect(6);

}

if (skillState < 34){

movespd = 0;

Sleep(10);

}

if ((x - (\*enemy).getX()) <= INIT\_HS\_RANGE || ((\*enemy).getX() - x) <= INIT\_HS\_RANGE){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-90); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemylhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

loadimage(player, "pic/rfourdash-3.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

x += 75; //加特技

loadimage(skillpic300, "pic/rfoursp-1.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 20 \* (skillState - 30), ox);

x -= 75;

}break;

case 3:{ //第2击

if (skillState == 30){

movespd = -150;

}

if (skillState < 29){

movespd = 0;

Sleep(30);

}

if ((x - (\*enemy).getX()) <= INIT\_HS\_RANGE || ((\*enemy).getX() - x) <= INIT\_HS\_RANGE){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-90); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemyrhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

loadimage(player, "pic/rfourdash-4.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

x -= 25;

y -= 20;

loadimage(skillpic300, "pic/rfoursp-2.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 20 \* (skillState - 25), ox);

x += 25;

y += 20;

}break;

case 4:{ //第3击

if (skillState == 25){

movespd = 150;

}

if (skillState < 24){

movespd = 0;

Sleep(30);

}

if ((x - (\*enemy).getX()) <= INIT\_HS\_RANGE || ((\*enemy).getX() - x) <= INIT\_HS\_RANGE){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-90); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemylhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

loadimage(player, "pic/rfourdash-5.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

y -= 30;

x += 13;

loadimage(skillpic300, "pic/rfoursp-3.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 20 \* (skillState - 20), ox);

y += 30;

x -= 13;

}break;

case 5:{ //第4击

if (skillState == 20){

movespd = -150;

}

if (skillState < 19){

movespd = 0;

Sleep(30);

}

if ((x - (\*enemy).getX()) <= INIT\_HS\_RANGE || ((\*enemy).getX() - x) <= INIT\_HS\_RANGE){ /\*\*\*\*击中判断\*\*\*\*/

(\*enemy).setHp((\*enemy).getHp() - (2 + logf(level))\*attack\*0.2); //击中造成伤害

(\*enemy).setStill(-90); //击中造成僵直

(\*enemy).setSpd(0); //击中造成僵直

if ((\*enemy).getType() == 9){

loadimage(enemyplayer, "pic/enemyrhit.jpg"); //加载被击中时姿势

MAINFRAME::M\_putimg((\*enemy).x, (\*enemy).y, enemyplayer, WHITE, 100, ox); /\*\*\*\*击中判断结束\*\*\*\*/

}

}

loadimage(player, "pic/rfourdash-6.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

loadimage(skillpic300, "pic/rfoursp-4.jpg", 340, 185, true);

MAINFRAME::M\_putimg(x, y, skillpic300, WHITE, 20 \* (skillState - 15), ox);

}break;

case 6:{ //收尾

loadimage(player, "pic/rfourdash-6.jpg", 354, 185, true);

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

if (skillState > 0 && (skillState < 10))

loadimage(skillpic300, "pic/foursp-5-6.jpg", 340, 185, true);

if (skillState>3 && (skillState < 10))

loadimage(skillpic300, "pic/foursp-5-4.jpg", 340, 185, true);

if ((skillState>6) && (skillState < 10))

loadimage(skillpic300, "pic/foursp-5.jpg", 340, 185, true);

MAINFRAME::M\_putimg((\*enemy).getX() - 110, y, skillpic300, WHITE, 6 \* skillState, ox);

if (skillState == 1){

combo = 0;

x += 110;

}

if (skillState == 1 || skillState == 0){

loadimage(player, "pic/lstill.jpg", 180, 185, true);

dir = 0;

}

MAINFRAME::M\_putimg(x, y, player, WHITE, 100, ox);

}break;

}

}break;

}

}break;//技能2结束

}

}

void PLAYER::startRun()

{

isRun = true;

}

void PLAYER::coolingDown()

{

coolDown--;

if (coolDown < 0)

coolDown = 0;

}

bool PLAYER::coolingJudge()

{

if (coolDown > 0)

return true;

return false;

}

float PLAYER::getMaxHp()

{

return maxhp;

}

void PLAYER::AIFind(int \*xf)

{

if (type != 10){

if (\*xf - x <= INIT\_FINDRANGE && \*xf - x > 100){

dir = 1; //改变方向

still = -5; //开始跑动

runState = 1; //使runState为1

movespd = 5;

isRun = true;

}

if (x - \*xf <= INIT\_FINDRANGE && x - \*xf > 100){

dir = 0; //改变方向

still = -5; //开始跑动

runState = 1; //使runState为1

movespd = -5;

isRun = true;

}

if (x == \*xf){

movespd = 0;

still = 1;

runState = 0;

}

}

if (type == 10){

if (\*xf - x <= INIT\_FINDRANGE && \*xf - x > 100){

dir = 1; //改变方向

}

if (x - \*xf <= INIT\_FINDRANGE && x - \*xf > 100){

dir = 0; //改变方向

}

}

}

void PLAYER::AIAttack(PLAYER \*player,IMAGE \*enemy)

{

if (((\*player).getY() - y<100 && (\*player).getY() - y>0) || (y - (\*player).getY()<100 && y - (\*player).getY() >= 0)&&run==0)

switch (type)

{

case 1:

case 2:

case 9:{

switch (dir){

case 0:{

loadimage(enemy, "pic/enemylattack.jpg");

if ((x - (\*player).getX()) <= INIT\_MELEEATTACK\_RANGE && (x - (\*player).getX()) >= 0){ /\*\*\*\*击中判断\*\*\*\*/

(\*player).setHp(((\*player).getHp() - attack)); //击中造成伤害

(\*player).setStill(-15); //击中造成僵直

(\*player).setSpd(-5); //击中造成击退

}

}break;

case 1:{

loadimage(enemy, "pic/enemyrattack.jpg");

if (((\*player).getX() - x) <= INIT\_MELEEATTACK\_RANGE && ((\*player).getX() - x) >= 0){ /\*\*\*\*击中判断\*\*\*\*/

(\*player).setHp(((\*player).getHp() - attack)); //击中造成伤害

(\*player).setStill(-15); //击中造成僵直

(\*player).setSpd(5); //击中造成击退

skillType = 1;

}

}break;

}

still = -20;

}break;

case 10:{

skillType = 1;

skillState = 50;

coolDown = 50;

//攻击判定

combo++;

still = -50;

soundEffect(9);

switch (dir){

case 0:{

if ((x - (\*player).getX()) <= 2\*INIT\_MELEEATTACK\_RANGE && (x - (\*player).getX()) >= 0){ /\*\*\*\*击中判断\*\*\*\*/

(\*player).setHp((\*player).getHp() - attack); //击中造成伤害

(\*player).setStill(-15); //击中造成僵直

(\*player).setSpd(-10);

(\*player).setRun(0); //击中造成击退

/\*\*\*\*击中判断结束\*\*\*\*/

}

switch (combo){

case 1:{

loadimage(enemy, "pic/GEattack1-1.jpg",425,425,true);

}break;

case 2:{

loadimage(enemy, "pic/GEattack1-2.jpg",567,425,true);

}break;

case 3:{

loadimage(enemy, "pic/GEattack3-1.jpg",425,510,true);

}break;

case 4:{

loadimage(enemy, "pic/GEattack3-2.jpg",510,425,true);

}break;

}

if (combo == 4)combo = 0;

}break;

case 1:{

if (((\*player).getX() - x) <= 2\*INIT\_MELEEATTACK\_RANGE && ((\*player).getX() - x) >= 0){ /\*\*\*\*击中判断\*\*\*\*/

(\*player).setHp((\*player).getHp() - attack); //击中造成伤害

(\*player).setStill(-15); //击中造成僵直

(\*player).setSpd(-10);

(\*player).setRun(0); //击中造成击退

/\*\*\*\*击中判断结束\*\*\*\*/

}

switch (combo){

case 1:{

loadimage(enemy, "pic/GErattack1-1.jpg", 425, 425, true);

}break;

case 2:{

loadimage(enemy, "pic/GErattack1-2.jpg", 567, 425, true);

}break;

case 3:{

loadimage(enemy, "pic/GErattack3-1.jpg", 425, 510, true);

}break;

case 4:{

loadimage(enemy, "pic/GErattack3-2.jpg", 510, 425, true);

}break;

}

}break;

}

if (combo == 4)combo = 0;

}break;

}

}

void PLAYER::levelUp()

{

level++;

statRefresh();

hp = maxhp;

}

void PLAYER::statRefresh()

{

attack = level \* 20+30;

maxhp = level \* 250;

hpregen = level\*0.1;

}

int PLAYER::getExp()

{

return exp;

}

int PLAYER::getLev()

{

return level;

}

void PLAYER::setExp(int num)

{

exp = num;

}

void PLAYER::expAdd(int num)

{

exp += num;

}

void PLAYER::levelUpCheck()

{

if (exp >= 25\*level\*level + level \* 5 + 20)

levelUp();

}

int PLAYER::getSkillType()

{

return skillType;

}

void PLAYER::hpRege()

{

hp += hpregen;

}

void PLAYER::respawn()

{

exp = exp\*0.9;

if (exp < 25 \* (level - 1)\*(level - 1) + (level - 1) \* 5 + 20)

exp = 25 \* (level - 1)\*(level - 1) + (level - 1) \* 5 + 20;

attack = level \* 20 + 30;

maxhp = level \* 250;

hp = maxhp;

hpregen = level\*0.1;

y = 400;

soundEffect(7);

}

void PLAYER::skillStateMove()

{

skillState--;

}

int PLAYER::getType()

{

return type;

}

void PLAYER::soundEffect(int soundtype)

{

if (SOUND)

switch (soundtype){

case 0:{

PlaySound("sound/boot.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 1:{

PlaySound("sound/press.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 2:{

PlaySound("sound/melee\_1.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 3:{

PlaySound("sound/welcome.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 4:{

PlaySound("sound/tele.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 5:{

PlaySound("sound/jump.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 6:{

PlaySound("sound/four.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 7:{

PlaySound("sound/res.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 8:{

PlaySound("sound/xiu.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

case 9:{

PlaySound("sound/boom.wav", NULL, SND\_FILENAME | SND\_ASYNC);

}break;

default:break;

}

}

bool PLAYER::getSound()

{

return SOUND;

}

//初始化用函数

void PLAYER::teleport(int x1, int y1) //用于玩家传送

{

x = x1;

y = y1;

soundEffect(4);

}

void PLAYER::setHp(float num)

{

hp = num;

}

void PLAYER::setAttack(int num)

{

attack = num;

}

void PLAYER::setMaxHp(float num)

{

maxhp = num;

}

void PLAYER::setSpd(int num)

{

movespd = num;

}

void PLAYER::setDir(bool num)

{

dir = num;

}

void PLAYER::setCombo(int num)

{

combo = num;

}

void PLAYER::setSkillState(int num)

{

skillState = num;

}

void PLAYER::setSkillType(int num)

{

skillType = num;

}

void PLAYER::setStill(int num)

{

still = num;

}

void PLAYER::setRunState(int num)

{

runState = num;

if (num > 0)

isRun = true;

}

void PLAYER::setRun(int num)

{

run = num;

if (num == 0)

isRun = 0;

}

void PLAYER::setLevel(int num)

{

level = num;

}

void PLAYER::setType(int num)

{

type = num;

}

void PLAYER::setSound(bool a)

{

SOUND = a;

}

EOF