**高级语言程序设计**

**课设报告**

**题 目\_\_\_\_\_\_\_\_小鱼小鱼快长大\_\_\_\_\_\_**

**学 号\_\_\_\_\_\_\_\_15143103 \_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_刘方瑞 \_\_\_\_\_\_\_\_\_\_\_**

**指导教师\_\_\_\_\_\_\_\_周珺 \_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**提交日期\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**成绩评价表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **报告内容** | **报告结构** | **报告图表** | **界面** | **最终成绩** |
| **□丰富正确**  **□正确**  **□基本符合要求**  **□过于简单** | **□完全符合要求**  **□基本符合要求**  **□有欠缺** | **□符合规范**  **□基本符合规范**  **□有一些错误** | **□丰富、炫酷**  **□有背景图片**  **□有背景色**  **□简单**  **□开始界面 □游戏界面** |  |
| **程序功能实现** | **程序执行情况** | **问题回答情况** | **总体评价** | |
| **□多个扩展功能**  **□少量扩展功能**  **□完成基本功能**  **□未完成基本功能** | **□流畅**  **□界面有闪动**  **□操作不灵活**  **□有错误** | **□清楚、正确**  **□基本正确**  **□回答有部分错误**  **□不能回答问题** |  | |

**教师签字:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

目录

[1 需求分析 3](#_Toc480034196)

[1.1 功能 3](#_Toc480034197)

[1.1.1 基本功能 4](#_Toc480034198)

[1.1.2 扩展功能 4](#_Toc480034199)

[1.2数据 5](#_Toc480034200)

[1.2.1 影响游戏难度的参数 5](#_Toc480034201)

[1.2.2 需要保存的内容 5](#_Toc480034202)

[1.2.3 游戏环境相关 5](#_Toc480034203)

[1.3界面 5](#_Toc480034204)

[1.4 开发与运行环境需求 6](#_Toc480034205)

[2 总体设计 7](#_Toc480034206)

[2.1 主要数据结构 7](#_Toc480034207)

[2.1.1 游戏结构体 7](#_Toc480034208)

[2.1.2 鱼节点结构 7](#_Toc480034209)

[2.2 程序模块设计 8](#_Toc480034210)

[2.2.1 子模块设计 8](#_Toc480034211)

[（1）开始游戏 9](#_Toc480034212)

[（2）继续游戏 9](#_Toc480034213)

[（3）选项 10](#_Toc480034214)

[（4）关于 10](#_Toc480034215)

[（5）退出 10](#_Toc480034216)

[2.2.2 函数功能 10](#_Toc480034217)

[（1）鱼的管理 10](#_Toc480034218)

[（2）游戏内容 11](#_Toc480034219)

[（3）接待画面 12](#_Toc480034220)

[（4）选项画面 12](#_Toc480034221)

[（5）菜单画面 12](#_Toc480034222)

[（6）文件操作 13](#_Toc480034223)

[（7）关于画面 13](#_Toc480034224)

[3 关键技术的实现 14](#_Toc480034225)

[3.1 动画的实现 14](#_Toc480034226)

[3.2 鱼的运动方式 15](#_Toc480034227)

[3.3 鱼的生成方式 15](#_Toc480034228)

[3.4 游戏的状态存取 16](#_Toc480034229)

[3.5 鱼的追踪/躲避算法 16](#_Toc480034230)

[3.6 渐变特效 17](#_Toc480034231)

[3.7 图片渲染 18](#_Toc480034232)

[4 测试 19](#_Toc480034233)

[5 总结提高 20](#_Toc480034234)

[5.1 课程设计总结 20](#_Toc480034235)

[5.2 对本课程意见与建议 21](#_Toc480034236)

[5.3 附件：程序源代码 21](#_Toc480034237)

# 1 需求分析

这次课设的目标是制作一个大鱼吃小鱼的游戏。主要要实现的功能有要实现小鱼在碰到高级的大鱼的时候要扣分，而在碰到比自己级数略低的小鱼时，要做出相应的奖励。在这个过程中，要对玩家的游戏成绩进行记录。这个程序应该有自己的菜单，以及游戏内容等等。

由于老师有严格限制语言和使用的函数库。所以我尽量使用最基础的函数完成整个游戏。在这一点上，可以从我的代码中完整的体现出来。

## 1.1 功能

游戏功能主要分为以下内容：

1. **游戏主菜单：**
   1. **开始游戏**

这里是玩家开始游戏的地方。

* 1. **继续游戏**

这里玩家可以恢复保存的游戏

* 1. **游戏选项**

这里玩家可以设定游戏的设定

* 1. **关于**

这里玩家可以看到关于作者与开发过程的内容

* 1. **退出**

这里玩家可以退出游戏

1. **游戏内容**
   1. **中断游戏**

玩家中断游戏，返回至主菜单

* 1. **渲染**

渲染游戏内容，我们采取了帧为单位的渲染方式

* 1. **判定**

判定玩家位置信息，并作出逻辑判断

* 1. **存储操作**

存储游戏相关数据结构（例如鱼的链表，以及鱼素材的纹理）

1. **游戏资料存储**
   1. **游戏内容存储**

将中断游戏时的游戏内容存储至文件，并可以从文件中恢复游戏内容。方便玩家进行游戏。

* 1. **游戏设置存储**

将游戏设置存至文件，包括分辨率设置（需重启游戏）以及难度设置。

### 1.1.1 基本功能

1、 小鱼可以通过鼠标移动

2、 小鱼可以吃掉比自己小的小鱼，同时得分（分数加一）

3、 小鱼在碰到比自己大的小鱼时，会受伤（分数减五）

4、 小鱼在吃掉或者受伤时都会改变自己大小（吃掉加一级，受伤减五级）

5、 小鱼可以使用清空“技能”（摁下C键减分但可以使屏幕内小鱼清空）

6、 小鱼可以吃掉随机位置出现的珍珠（吃掉后分数加一百）

7、 比玩家级别高的小鱼可以向玩家发起攻击（追踪）产出围困的效果

8、 比玩家级别低的小鱼可以躲避玩家

### 1.1.2 扩展功能

1、 游戏添加了更加美观的开场渐变动画，显示游戏作者的logo

2、 游戏拥有更加美观并且易于操作的菜单。（仅键盘）

3、 游戏添加了读取并恢复中断游戏的功能，玩家在游戏中敲击Esc后游戏会被存储为存档，以供下一次玩家选定Resume选项恢复。

4、 游戏加入了设置界面，通过设置界面玩家可以选择不同的不同难度以及游戏的不同分辨率选项。（分辨率选项需要重启游戏）

5、 游戏本体仅为一个可执行文件，方便拷贝以及运行，向下兼容XP系统

6、 优化了渲染流程，使游戏体验更加流畅

## 1.2数据

### 1.2.1 影响游戏难度的参数

1）游戏模式 （整型）

2）速度比率 （单精度浮点）

### 1.2.2 需要保存的内容

1. 玩家名字 （字符串）
2. 起始时间戳，结束时间戳 （TIME）
3. 游戏得分 （整型）
4. 无敌模式 （布尔值）
5. 游戏级别 （浮点）
6. 电脑NPC链表头 （FISH结构体指针）

### 1.2.3 游戏环境相关

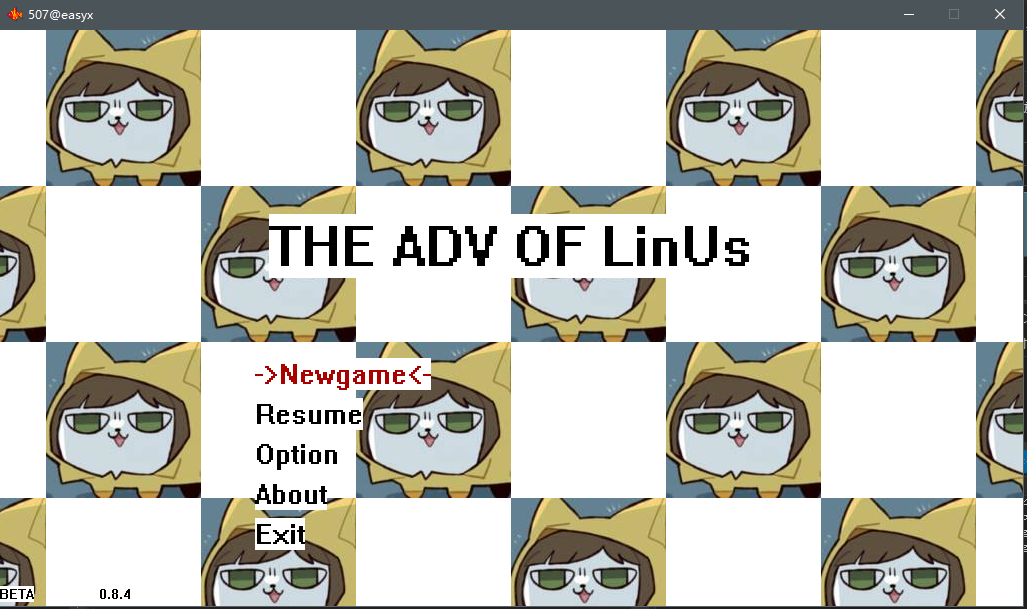
1）鼠标消息 （MOUSEMSG）

2）NPC图片素材表 （IMAGE[]）

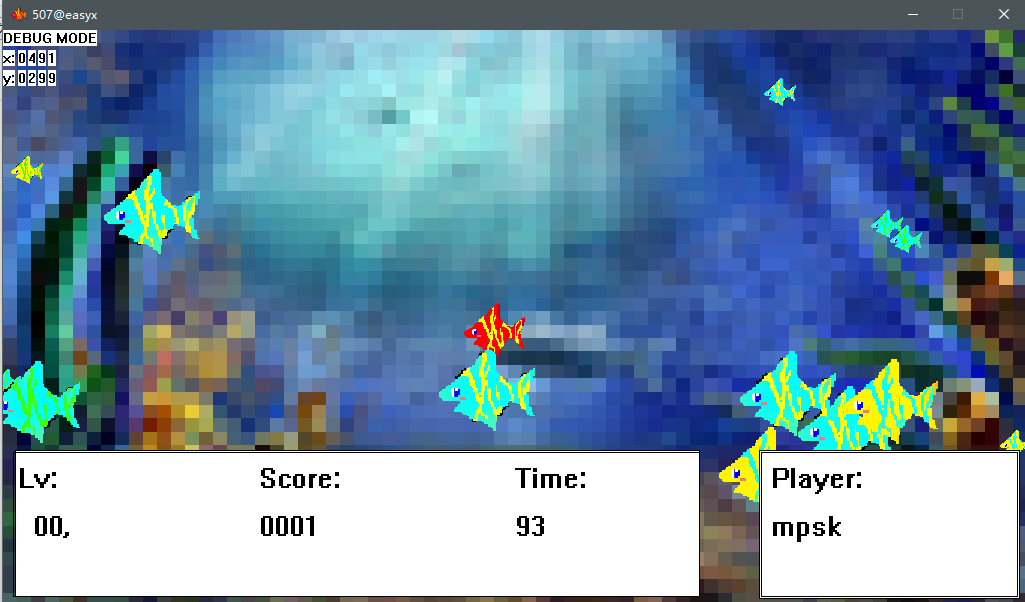
3）玩家图片素材 （IMAGE）

4）背景图片素材 （IMAGE）

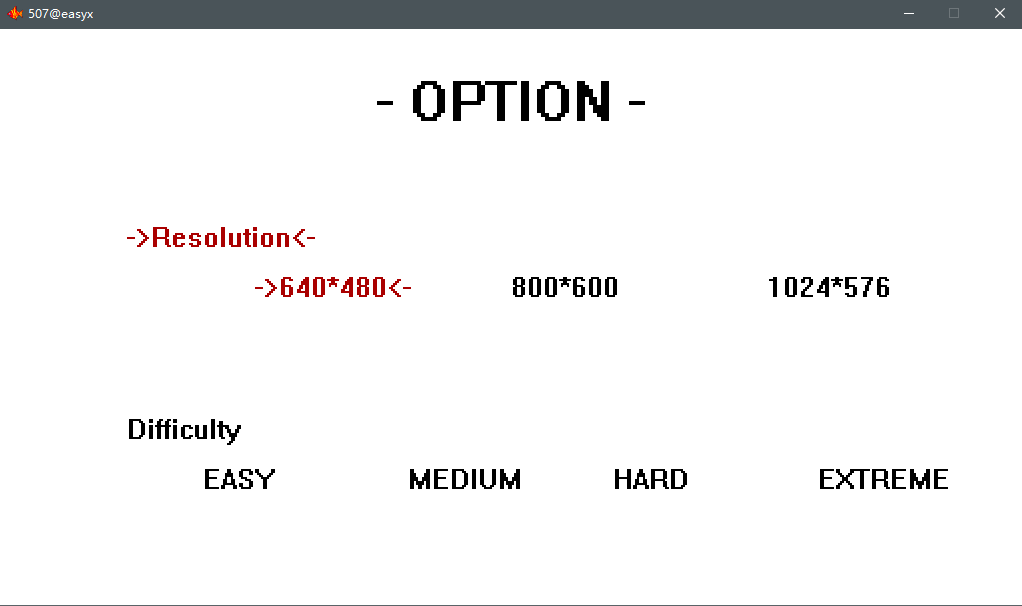
## 1.3界面



游戏主菜单



游戏内容



游戏设置界面

## 1.4 开发与运行环境需求

开发环境：Visual Studio 2015

源代码以及工程文件中包括了引用的EasyX静态库文件以及头文件

CPU： 1.6GHz Pentuim 以上 推荐 2.4GHz 赛扬处理器

Memory： 1GB以上 推荐 2GB

运行系统： Windows XP SP1以上 推荐 Windows 7 SP1

分辨率： 640\*480以上 推荐 1024\*576

# 2 总体设计

在需求分析的基础上，采用自顶向下的方法进行模块化的程序设计，合理划分模块。在此部分应给出程序总体的设计方案，划分了几个模块。

## 2.1 主要数据结构

### 2.1.1 游戏结构体

游戏结构体GAME，包含游戏内的所有参数，以及进行内容操作的资源。

typedef struct game

{

    //  mode 0 for rand position based

    //  mode 1 for rand speed based

    //  other is for rand speed without y

    int         mode = 2;

    int         speed\_ratio = 1;

    wchar\_t     \*player;                    //  Player's name

    int         time\_begin, time\_sec;       //  Timer for count

    int         score = 0;                  //  Score

    bool        god;                        //  GOD Flag

    float       level;                      //  Player's level

    MOUSEMSG    mouse;                      //  Mouse Message

    IMAGE       npc\_fishes[RES\_FISHES];     //  NPC Texture Array

    IMAGE       player\_fish;                //  Player's fish texture

    IMAGE       background;                 //  Background Texture

    FISH        \*fish = NULL;               //  Head of NPC chain list

}GAME;

### 2.1.2 鱼节点结构

鱼节点结构：用来存储鱼类特征的结构体，鱼的存储方式是线性链表

typedef struct fish

{

    float x;                //  Node fish's position in X

    float y;                //  Node fish's position in Y

    float s\_x;              //  Node fish's speed in X

    float s\_y;              //  Node fish's speed in Y

    float level;            //  Node fish's level(judge score or hurt)

    int res\_num;            //  Alternate texture amount

    int flag;               //  From Left or Right

    fish \*next;             //  Pointer to next

}FISH;

这里我们抽象了每个NPC的位置与速度。我们不直接操控NPC的位置（不直接让位置参与递增或递减运算），而是通过控制抽象速度来控制鱼的运动。鱼的位置是通过将鱼的速度计算得来的。

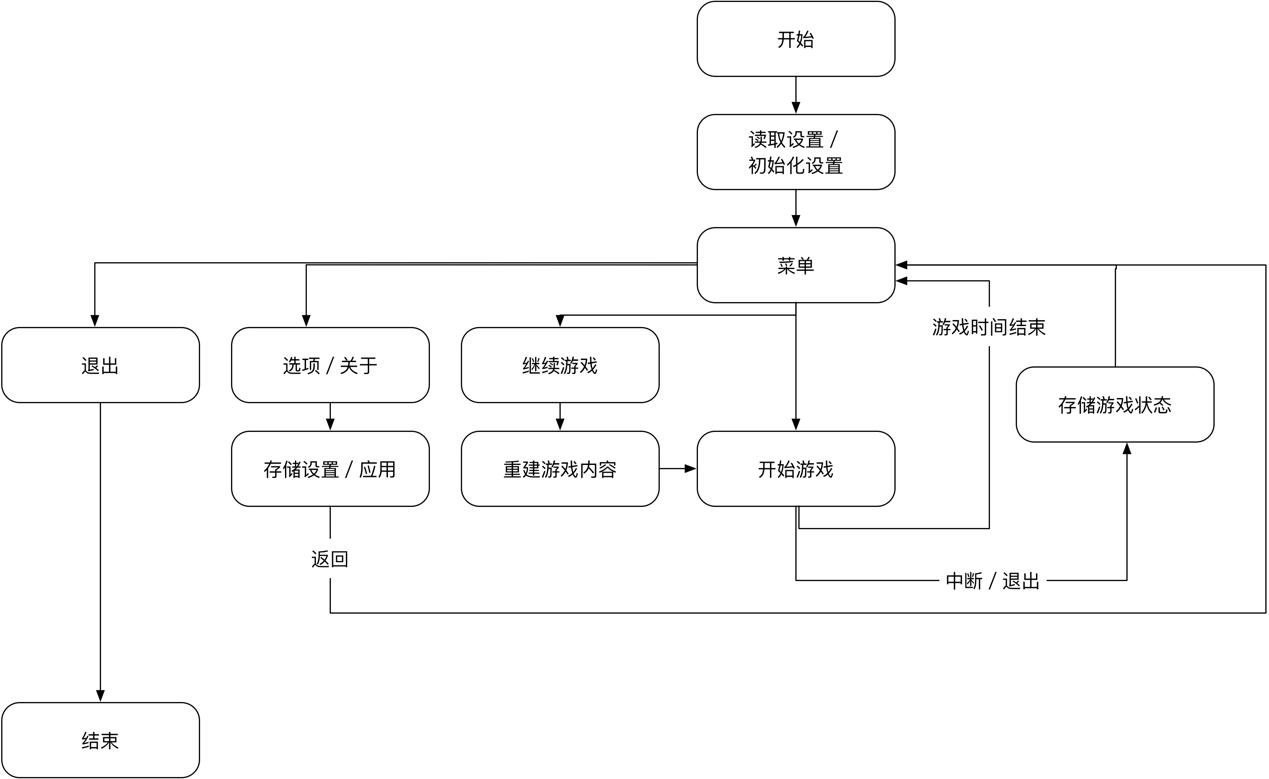
鱼的速度是通过随机方法得来的。它在y方向的速度是完全的随机，但是x方向的速度是有偏向性的，这使得不同朝向的NPC可以朝向对应的方向运动。

我在游戏世界中抽象了阻力系数。这样鱼类在运动的时候速度就会慢慢减小直至接近停止（并不是绝对的等于停止，而是存在阈值控制）。在近似停止的情况下，程序会随机为NPC生成一个速度，让NPC继续运动下去。这部分我们会在下面继续详细解释。

同时，为了防止鱼在同一个地方卡住，我们还规定了flag，来区分不同来源的NPC，让他们能够大致朝向同一方向运动。

为了增加NPC的多样性，我们使用了NPC图片素材表的下标值来标记NPC不同的纹理。而且我们还可以通过随机数来让NPC使用不同的纹理。

## 2.2 程序模块设计



### 2.2.1 子模块设计

我接下来的所有程序设计都是围绕着操作游戏结构体来展开的。实际上这些模块都是逻辑上的，仅仅体现在函数命名上。

#### （1）开始游戏

这个模块主要是创建一个新的游戏结构体并维护。使其完成对游戏内容的更新与渲染。此模块中主要的内容就是维护一个游戏结构体。

在加入存储功能以后，这个模块被拆分成了建立游戏对象和维护游戏对象两个部分。

首先我们在进入游戏的时候，会询问玩家的名字，这个时候程序会组建游戏结构体的内容，初始化所有的游戏变量，包括NPC链表、玩家级别、素材的载入。

之后程序会提示准备好，玩家就将进入游戏界面开始游戏。

开始游戏后，玩家碰到级别更小的鱼会分数加一，而碰到级别更大的鱼则会扣掉5分，同时玩家会获得大概1秒左右的无敌。在无敌状态下，玩家不会受到伤害。（分数不会减少）

游戏如果在游戏时间结束前按下了ESC，那么就认为玩家中途退出，程序会自动将当前游戏状态（包括游戏级别，游戏得分，游戏NPC的所有信息）存储至文件，使之可以在下次运行（可以关闭游戏）时，恢复游戏的内容。

当然在游戏结束的时候（不管是中途退出还是游戏时间结束），玩家的得分都会显示在屏幕上。当然在游戏时间结束后，玩家是没有办法再次存档的。

#### （2）继续游戏

这个模块主要是从文件中恢复之前存储的游戏内容，相当于从文件读取游戏内容并重新组织一个游戏结构体。

当玩家选择了这个选项时，程序会从唯一的存档文件中格式化读取之前程序存储的内容。

存储文件的起始是玩家的名字，玩家的得分，玩家的级别，以及玩家的游戏时间。这些相对静态的游戏内容在存档文件中很容易恢复。但是对于NPC链式表，就需要格式化读入后，动态再次组建。由于链表中元素都相对平等，于是他们之间的物理存储位置并不影响游戏内容的再现。因此，我们就可以根据影响NPC行为与表现的信息重新组织结构体，并将他们再次相连。在组装好链表后，程序会将组织好的链表头返回给游戏结构体。这样一个游戏结构体就重新组装好了。

重组好了游戏结构体后，我们就需要将游戏结构体重新送回维护函数，进行正常的游戏内容了。

当然如果玩家从恢复的游戏过程中中途退出，其进度一样会被保存。

#### （3）选项

这个模块用于维护从文件中读取或向文件中写入设置，并将设置传入主程序。

玩家可以选择不同的分辨率，但是这个操作需要重新启动。因为我们在创建绘图区域的时候，就需要一个固定的值。我们在选项中提供了三个常用的分辨率，当然玩家也可以通过自己修改Setting.bin文件来自己设定分辨率参数，来获得更好的体验。但是这种操作可能会带来任何不可意料地后果。例如画布创建失败等等。一旦出现这种情况，仅仅需要删除Setting.bin，清空已经保存的设置文件，程序就可以重新生成正确参数的设置文件。

关于难度，目前的版本我抽象了两个参数，一个是mode，另一个是speed\_ratio。这两个变量分别控制鱼类的运动模式，和运动的修正因数。我认为通过控制这两个变量可以有效控制游戏的难度。游戏内置了四种预设难度，当然玩家也可以通过手工修改这两个参数，来达到生成自定义难度的游戏。但是同时和分辨率一样，手工修改可能会导致不可意料的后果，如果出现游戏崩溃的情况，只需要删除相应的设置文件并重启游戏就可以了。

#### （4）关于

关于页面介绍了我的个人信息，以及我想对玩家说的话。我的整个程序文件和开发进度，以及一些相关的信息可以在github上我的主页找到。我的github账号是mpskex@163.com。欢迎各位老师关注。

#### （5）退出

这个部分就是使游戏退出的。这里游戏所有的元素以及相关的变量都会被清空，并释放掉。

### 2.2.2 函数功能

#### （1）鱼的管理

本模块用于和鱼相关的操作。

相关函数：

int fish\_add(GAME &game, int num, int SCREEN\_W, int SCREEN\_H);

功能：向鱼链表中添加一条鱼

参数：游戏结构体，添加数量，屏幕宽，屏幕高

返回值 整型，表示添加状态（成功/失败）

int fish\_clear(GAME &game)

功能：清空鱼链表

参数：游戏结构体

返回值：整型，表示删除状态

int fish\_init(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：初始鱼链表，创建链表的头指针

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示删除状态

int fish\_judge(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：判定鱼的位置是否与玩家相交

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示判断状态（成功/失败）

FISH \*fish\_rm(GAME &game, FISH \*fish)

功能：删除特定的鱼

参数：游戏结构体，屏幕宽，屏幕高

返回值：指向鱼类型的指针（被删除的鱼的下一条鱼）

int fish\_single(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：鱼类显示的单周期函数，包括鱼类运动方式

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

#### （2）游戏内容

int game\_background\_single(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：显示游戏世界背景

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

int game\_loop(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：游戏世界渲染循环

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

int game\_main(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：游戏内容主函数

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

返回值决定了游戏走向（返回值代表退出游戏类型）

int game\_player\_single(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：显示玩家对象

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

int game\_score(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：显示最终游戏结果

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

int game\_status\_single(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：游戏状态栏显示

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

#### （3）接待画面

int name\_loop(GAME &game, int SCREEN\_W, int SCREEN\_H)

功能：接待画面渲染循环

参数：游戏结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

#### （4）选项画面

int option\_loop(OPTION option, int SCREEN\_W, int SCREEN\_H)

功能：选项画面渲染循环

参数：选项结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

#### （5）菜单画面

int menu\_loop(MENU &menu, int SCREEN\_W, int SCREEN\_H)

功能：菜单画面渲染循环

参数：菜单结构体，屏幕宽，屏幕高

返回值：整型，表示单步执行状态（成功\失败）

void menu\_show\_version(int SCREEN\_W, int SCREEN\_H)

功能：菜单页面显示版本

参数：屏幕宽，屏幕高

返回值：无

#### （6）文件操作

int load\_setting(SETTING &setting)

功能：从文件载入设置

参数：设置结构体

返回值：整型，表示单步执行状态（成功\失败）

int write\_setting(SETTING setting)

功能：向文件写入设置

参数：设置结构体

返回值：整型，表示单步执行状态（成功\失败）

int write\_save(SAVE \*save);

功能：向文件写入内容

参数：游戏状态结构体

返回值：整型，表示单步执行状态（成功\失败）

int load\_save(SAVE &save);

功能：从文件读取内容

参数：游戏状态结构体

返回值：整型，表示单步执行状态（成功\失败）

#### （7）关于画面

void about\_loop(ABOUT about, int SCREEN\_W, int SCREEN\_H);

功能：关于画面渲染循环

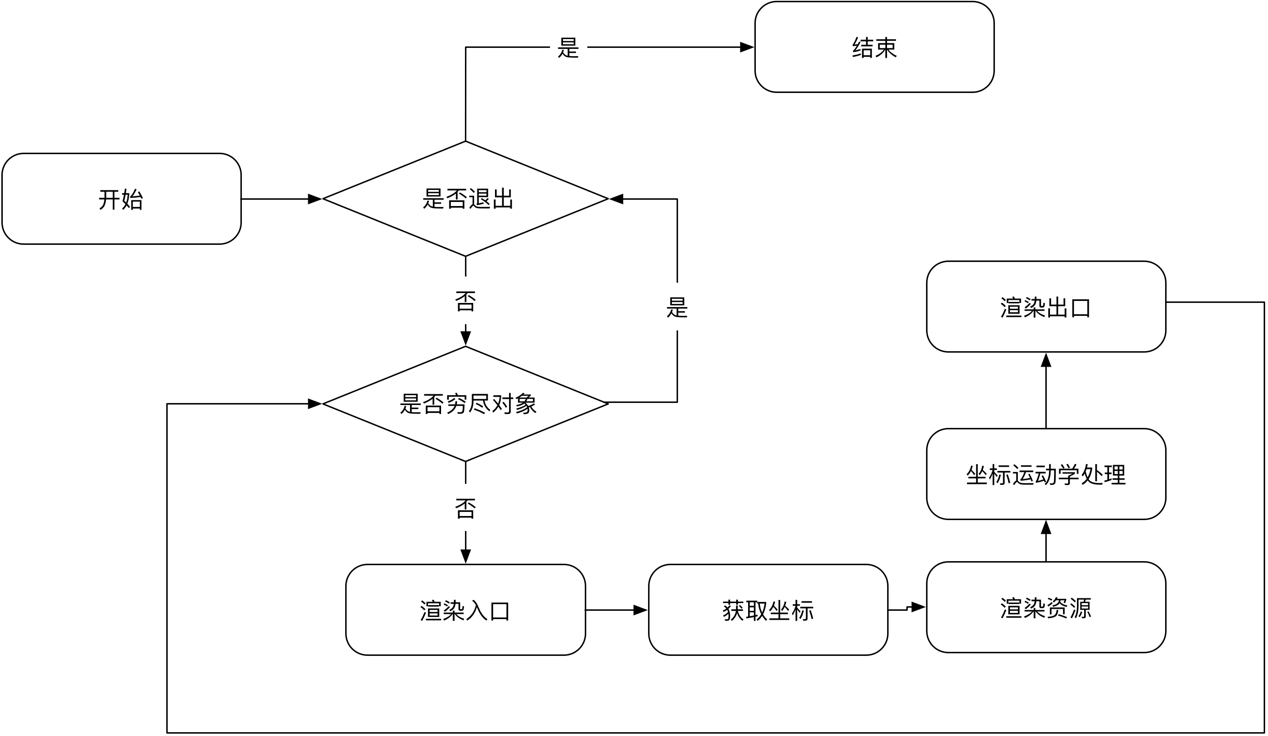
参数：关于结构体

返回值：无

# 3 关键技术的实现

这部分我们介绍了本程序一些实现上的特点。但是像一些基础技术，类似于链表的历遍以及删除追加，我这里就不再赘述。

## 3.1 动画的实现



历遍NPC链表的过程是整个游戏渲染Loop中的重中之重，因为这段代码是最容易出现指针越界，非法指针的。因此对于涉及到链表以及指针的代码我都是相当谨慎的。

在完成这部分代码的时候，其实比较顺利，主要就是注重起始条件，迭代过程，和终止条件。如果对于整个要操作的边界情况我们都考虑清楚的话，链表就会变得非常灵活而且看起来更加聪明伶俐。

我们在迭代渲染这部分内容的时候，会对于每一个NPC节点的位置信息进行计算并且将其位置用对应的素材画出来。当我们对于每一个节点都历遍的时候（判定条件指向指针等于空），我们就将之前绘制的内容呈现在屏幕上。由于计算机处理速度相当快，所以历遍链表并且绘制图像的速度非常短，可以说是一瞬，那么通过循环地迭代这个过程，再通过一定手段控制NPC的坐标，我们就可以绘制出流畅的动画了。

在将函数历遍所有的游戏资源以后，我们就将内容绘制在屏幕上。并且进入下一次迭代的开始，但是首先我们需要把游戏的画面清空，才可以进行下一次画面的绘制。

这里我们引入一个帧数的概念。我们在渲染画面的时候，每一个loop消耗的时间都是相似的，但是随着绘制对象的增多，比如鱼类的增加，绘制函数的消耗时间就会线性增加。那么给玩家的感官上的体验就会下降很多。因为随着绘制频率的下降，画面会出现明显的顿挫感，从而影响玩家的游戏体验。

那么这个问题大体通过两个方面解决。一个是通过提高绘图的效率（比如使用硬件加速绘图接口OpenGL等等）另一个就是通过稳定渲染帧数。（比如引入渲染队列以、锁帧、以及添加线性插值补间等等）我们这里采取了后者，使用的是锁帧的技术。由于我们的游戏中的抽象时间（比如鱼类与主角npc判定的时间，以及鱼类速度更新时间）是以帧数为基准的。帧数的降低是不仅会影响到游戏的画面，还会影响游戏的操作。因此使游戏帧数稳定是一个非常要紧的问题。

因此我们使用了WinAPI的多线程API创建一个线程为我们计时。这样做是因为线性执行导致的计时功能永远是Loop执行时间加上Sleep时间，达不到规范游戏时间的目的。

我们在Loop函数中加入了

wait = CreateThread(NULL, 0, game\_lock\_frame, NULL, 0, NULL);

其中game\_lock\_frame就是Sleep(time)函数，经过测试运行，验证了这个方法的确改善了游戏的整体体验。尤其是在画面和操控上。

## 3.2 鱼的运动方式

我们采用了抽象鱼的速度与鱼的位置，方便我们的坐标的运算。

根据公式△*r* =*r*（*t*+△t）－*r*（*t*），我们就可以用一个常量来表示一个固定的抽象加速度，然后根据随机生成的x，y速度，计算出对应的鱼的位置。

根据公式v = ∆S／∆t，我们可以将∆t简化为1（一个单周期），那么得到的了公式∆S = v，这样我们简易的运动学模拟就完成了。

鱼的运动系数主要是通过速度描述的，而这个速度还会乘上一个时间系数SPEED\_RATIO（宏）（在早期版本是宏形式出现的），相当于∆S = v \*（∆t \* speed\_ratio）。因为不直接操作位置，我们就能获得更佳平滑的NPC运动效果，也会显得更佳自然。

鱼的速度是通过随机方法得来的。它在y方向的速度是完全的随机，但是x方向的速度是有偏向性的，这使得不同朝向的NPC可以朝向对应的方向运动。

我在游戏世界中抽象了阻力系数。这样鱼类在运动的时候速度就会慢慢减小直至接近停止（并不是绝对的停止判定，而是存在阈值控制）。在近似停止的情况下，程序会随机为NPC生成一个速度，让NPC继续运动下去。

## 3.3 鱼的生成方式

如果NPC生成的大小都是固定的，那么游戏就没有办法推进。但如果鱼类生成的大小过于随机，那么也许会生成极小的鱼，影响游戏的观感与体验。那么如何生成大小合适的鱼类就成了一个非常需要解决的问题。

\_fish->level = rand() % 10 / 7 + game.level - (0.5 / game.level);

这就是控制生成鱼类大小的公式。首先这个随机数除以7，是在控制生成数的散度，如果这个散度过大，就也许会出现生成过小或者过大的鱼类。之后与游戏级别加和的部分是要将现在游戏级别作为生成鱼的基准。最后减去的部分是作为修正因数出现的。如果游戏级别过高，那么生成小鱼的几率就会变小，使得游戏的难度虽玩家级别的上升而增高，从而达到压制最高分的功能。

NPC出现的位置也是需要考虑的问题。鱼类应该是在屏幕两端生成。并且由于前面运动系统设计导致我们需要将两侧的鱼类分开，那么我们就是用了flag来区分。flag为1的鱼类沿x轴速度只会递增，flag为2的鱼类沿x轴的速度只会递减。但是在NPC生成的同时，我们需要给他一个初始速度，那么这个初始速度也是根据他们的flag值确定的。根据不两种flag，我们对应生成正与负的沿x轴的初始速度。这样的话他们就会规矩地向游戏场景中央移动了。

如果NPC的外观只有一种，那么就会显得太过于单调了。因此我加入了不同的NPC外观素材。在生成的过程中，我们使用了随机方法，为他们分配素材表中的对应下标，并在渲染函数过程中，提取对应下标的素材，完成不同外观的NPC的绘制。

## 3.4 游戏的状态存取

存储文件的起始是玩家的名字，玩家的得分，玩家的级别，以及玩家的游戏时间。这些相对静态的游戏内容在存档文件中很容易恢复。但是对于NPC链式表，就需要格式化读入后，动态再次组建。由于链表中元素都相对平等，于是他们之间的物理存储位置并不影响游戏内容的再现。因此，我们就可以根据影响NPC行为与表现的信息重新组织结构体，并将他们再次相连。在组装好链表后，程序会将组织好的链表头返回给游戏结构体。这样一个游戏结构体就重新组装好了。

重组好了游戏结构体后，我们就需要将游戏结构体重新送回维护函数，进行正常的游戏内容了。

我们使用将结构体的数值型内容（除了与结构体的头指针）转存入一个存档容器中（SAVE结构）然后我们分别将这个结构中的变量一一写入文件当中。这种存储方式可以以文本形式打开存档文件，而且内容还是可读的格式。（使用fprintf）

读取的时候我们需要重新根据文本文件中的内容重新组织链表。对应每一条记录，我们需要新建一个节点，并将它的上一个节点的子节点指针赋值。这样就非常的简单地完成了重建链表的功能。

## 3.5 鱼的追踪/躲避算法

我们的鱼类在获得随机的速度以后，大鱼并不能够有效的对玩家构成威胁，而小鱼又没有有效的避开玩家形成难度。这样一来游戏的可玩性就会大大降低。因此为了给游戏性带来一定提升，我们设计了一个能够对玩家构成一定威胁的大鱼追踪算法。

p->s\_x += SPD\_CHASE \* 0.001 \* (game.mouse.x - p->x);

p->s\_y += SPD\_CHASE \* 0.001 \* (game.mouse.y - p->y);

大鱼的速度导率是跟大鱼位置与玩家位置之差成正比，为了保证其符号，我们这里没有取平方。那么也就是说，玩家离鱼类越远，鱼类就会外加一个更大的速度来靠近玩家。因此，在玩家远离鱼群的时候，就会受到非常猛烈的攻击。

至于SPD\_CHASE，是为了方便对追踪速度导率进行补正的参数。而这个0.001是我们在测试时得出的合理的放大因数。在这个尺度下，鱼类的行为更加适合游戏的难度。

大鱼有了躲避算法，那么小鱼就该有躲避算法。我们的设计思路是小鱼的躲避速度应该随着玩家距离的减小而增大。因此我们首先想到了反比函数。因此我们设计了这样的躲避算法。

p->s\_x -= SPD\_EVADE \* 15 / (2 \* (game.mouse.x - p->x));

p->s\_y -= SPD\_EVADE \* 15 / (2 \* (game.mouse.y - p->y));

小鱼的速度导率是与玩家到小鱼距离成反比关系。这个满足了我们的设计。而且通过之后的测试修正，鱼类的行为表现良好，在逼近小鱼时，小鱼会出现轻微躲避的行为。

其实这样的算法有其缺陷，因为是在原有速度上进行补正，所以会出现npc在对玩家行为反应出现迟滞，导致其AI性能下降，但是这里我们就不再赘述这个算法的修改思路了。

## 3.6 渐变特效

我们在游戏的开始加入了一个我自己设计的logo，并对这个logo进行了渐变变换。由于这个渐变动画还是有一定的技术含量的，我就在这里说一下。

首先我们在逻辑上进行渐变变换的时候，我们其实就是在将图片变“亮”或者变“暗”。那么这个亮与暗就涉及到光照因数L。这个光照因数L的最大值我们称为Lmax。

由于easyX根本没有渐变的函数，也没有能够载入alpha通道的函数，所以我们就只能够自己实现。第一步我们应该能够历遍图片的每一个像素。首先使用DWORD\* pbImg = GetImageBuffer(&(intro.bk))得到图片的句柄。那么对于图片上的每一个像素，就可以用表达式：

(pbImg[i\*(intro.bk).getwidth() + j])

表示出来。由于图片像素的RGB值全部为整型，如果直接进行除法，一定会影响其数值，影响到图片质量。因此考虑到我们需要渐变的灰度阶，我们就定义Lmax等于64（2的六次方）。我们在最亮的部分应该使图片的RGB值等于图片原始值，在最暗的时候使绘制的RGB全部为0。

那么我们就总结出了：对于一幅图片I，其经过灰度修正系数L处理后的灰度修正图像I’满足下式：

I’ = I \* L (0 < L < 1)

根据上式，我们就可以对整型的RGB值与整型的灰度修正因数L与整型的修正因数最大值Lmax进行一下描述：

pbImg[i\*(intro.bk).getwidth() + j] \* L >> log2(Lmax)（1 < L < Lmax）

这样一来，我们就可以通过操纵L在不同时间绘制不同灰度修正因数的图像来达到渐变效果了。而渐变的效果完全可以定义不同的f(L)来实现不同的炫酷的效果。

根据上式，我们就可以实现对图片对象进行简易的渐变效果动画了。

## 3.7 图片渲染

在这次课设中我们使用了*GDI*中的*TransparentBlt()*函数。这个函数既可以完成去掉图片的边框的功能，又可以实现让素材以不同的尺寸渲染的功能。

首先我们先了解一下这个函数的原型

BOOL TransparentBlt(HDC hdcDest, //HDC目的设备句柄（我们的窗口）

int nXOriginDest, //目的设备上的目标位置x坐标

int nYOriginDest, //目的设备上的目标位置y坐标

int nWidthDest, //目标设备上绘图区域的宽

int hHeightDest, //目标设备上绘图区域的高

HDC hdcSrc, //HDC源设备句柄（这里是我们的图片）

int nXOriginSrc, //源设备上的源位置x坐标

int nYOriginSrc, //源设备上的源位置y坐标

int nWidthSrc, //源设备上的源位置的宽

int nHeightSrc, //源设备上的源位置的高

UINT crTransparent)；//替换掉的颜色（视作透明的颜色）

这个函数可以用于很多功能上。首先可以使用这个函数进行图片的渲染。*EasyX*在载入图片时其实创建了*GDI*的*HDC*设备，那么我们可以使用

*HDC GetImageHDC(IMAGE\* pImg = NULL);*

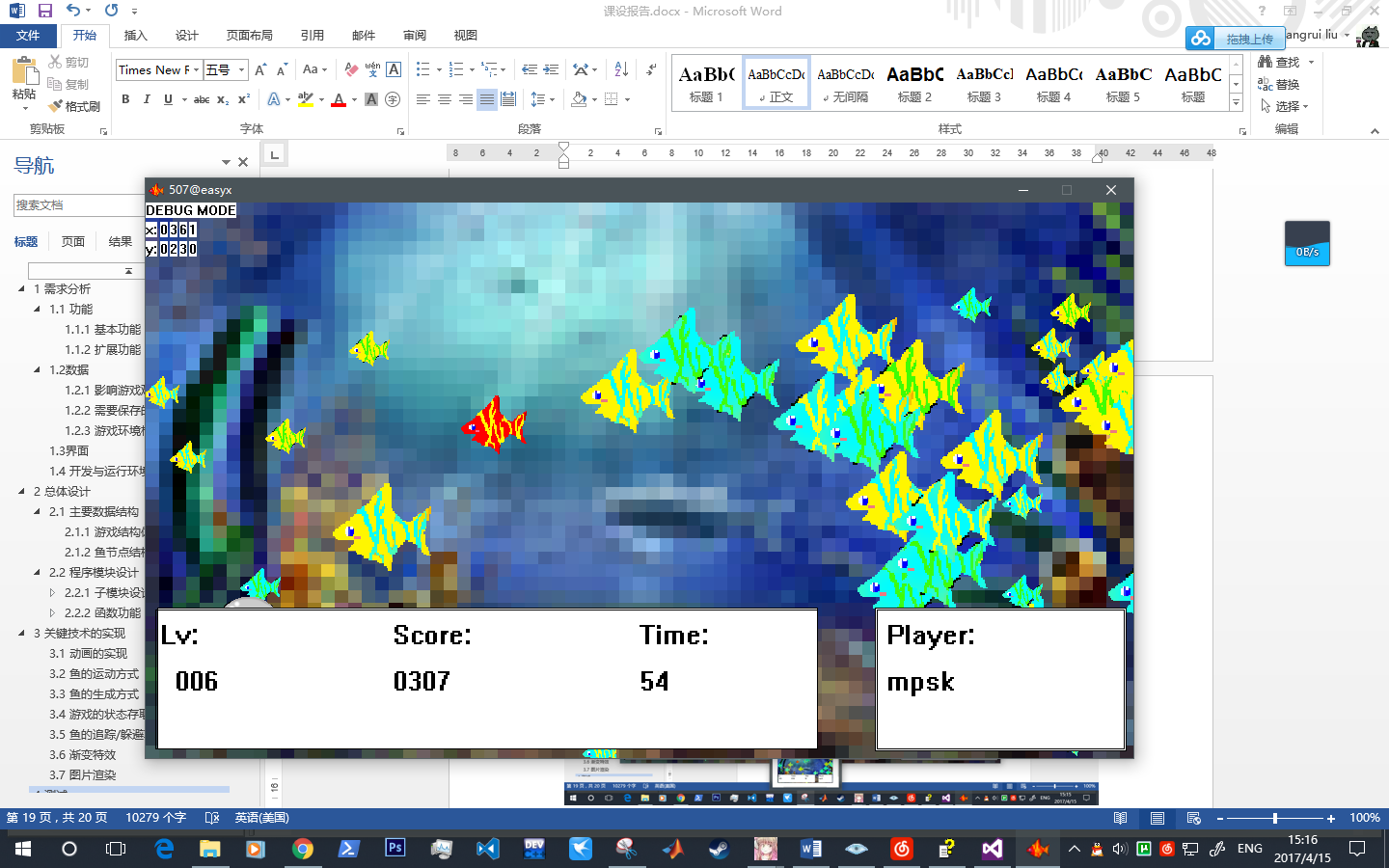
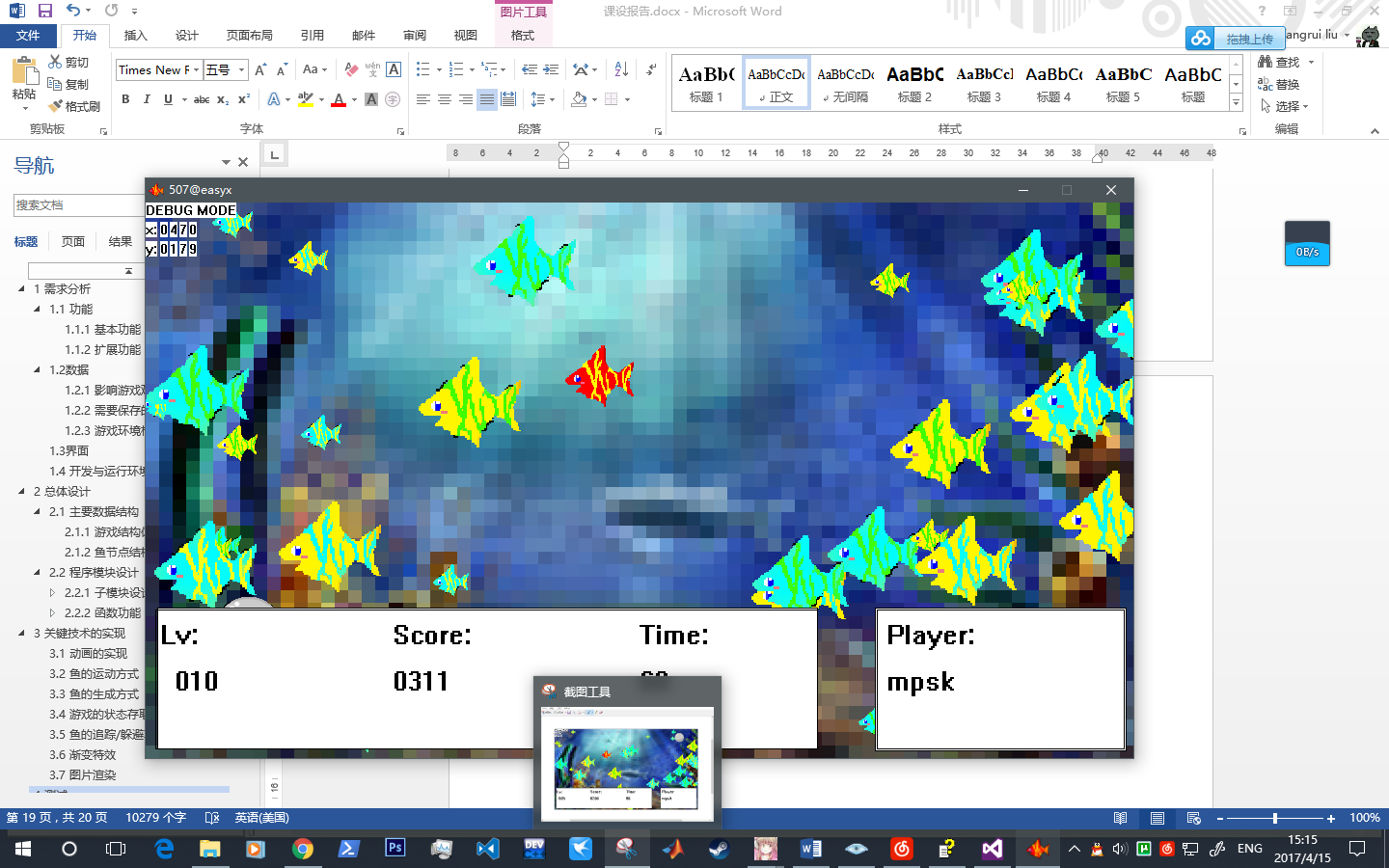
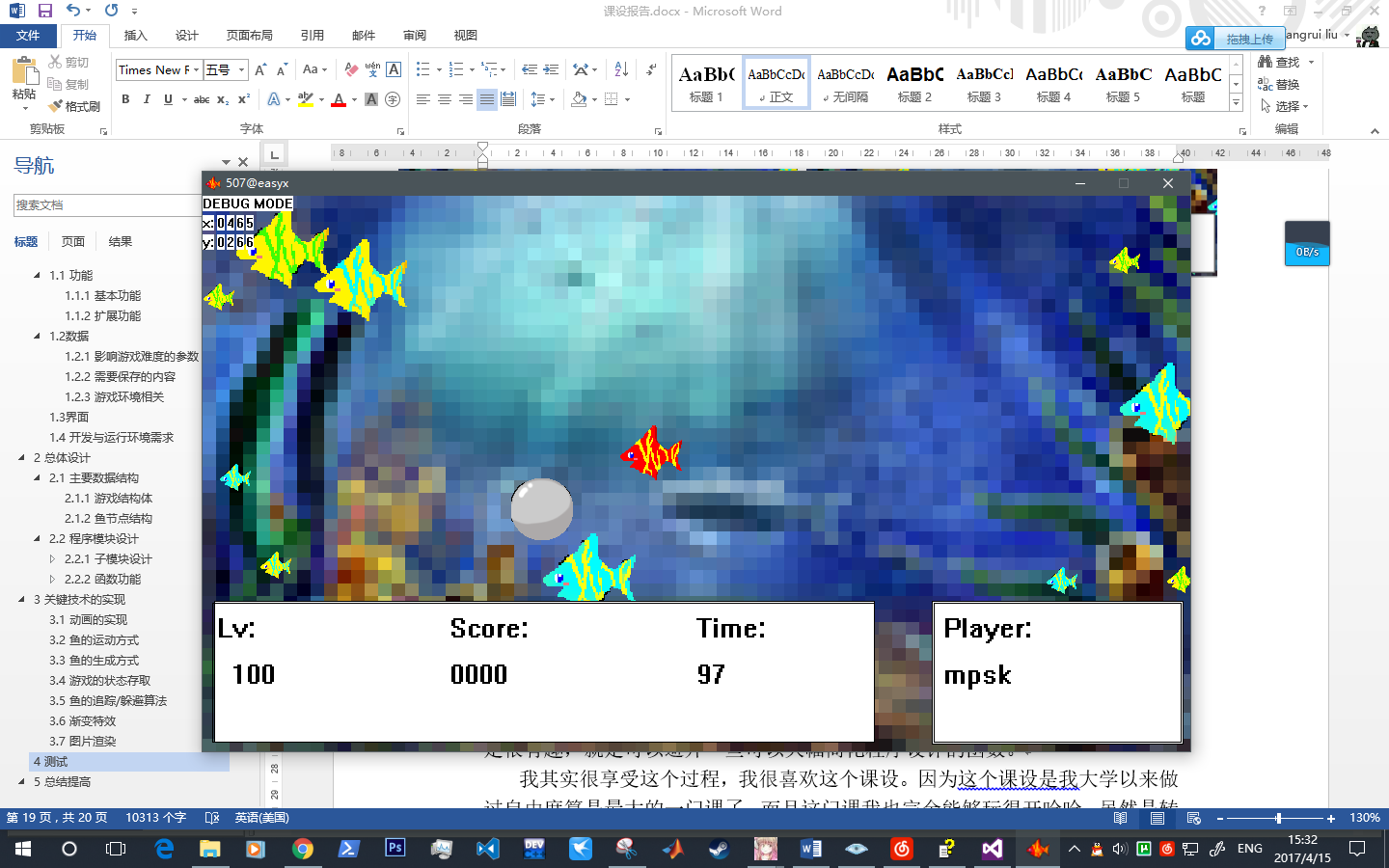
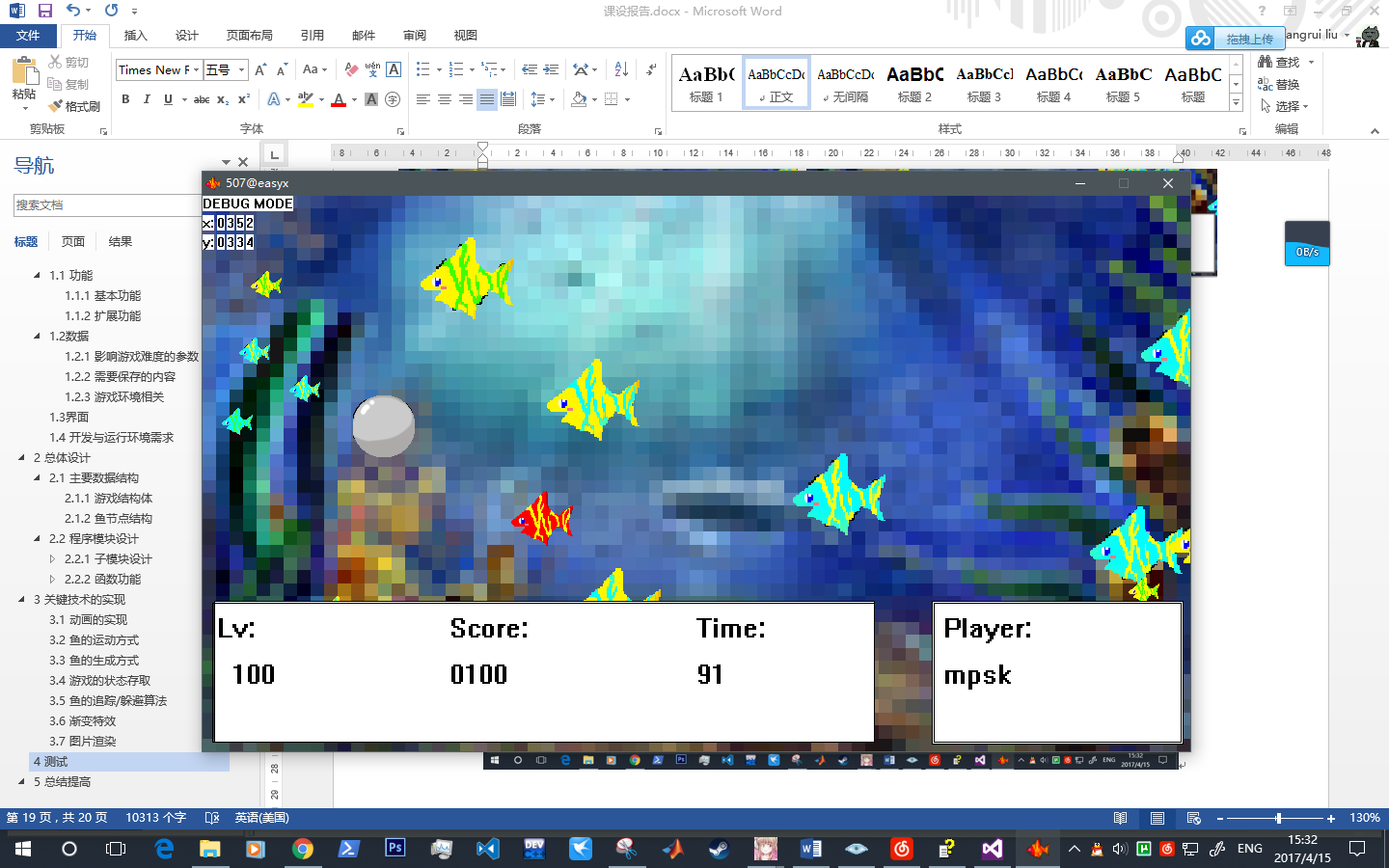
来获取该*IMAGE*对象和窗口对象的句柄。源设备上的源位置就使用（0,0，width,height），而在目标位置渲染的参数为（x，y，x + width，y + height）这样就可以完成最基本的等尺寸图像渲染。

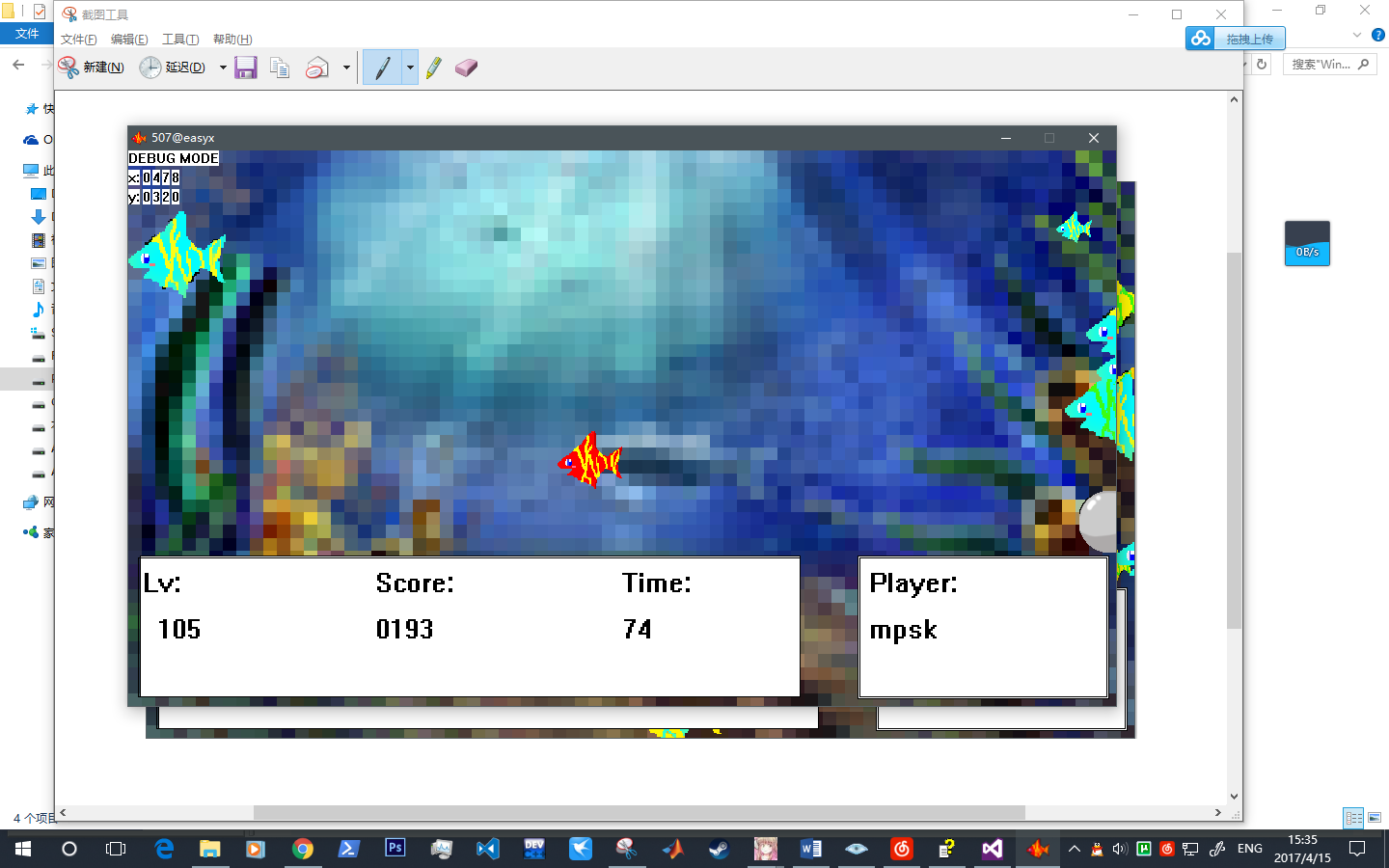
我们还可以通过改变渲染参数来实现等比渲染和不等比渲染。其实实现起来就只是改动目标位置的渲染参数。拿等比渲染来举例，其目标参数为（x，y，x + SCALE \* width，y + SCALE \* y）这里的SCALE就是我们的比例参数。通过调整SCALE就可以渲染出不同大小的等比图像了。

同时我们也可以通过使用控制待绘制的源的位置来选择不同的部分绘制。这个技术可以应用在元素动画上。对于同一个元素，绘制它的不同位置来达到使其在屏幕上体现其流畅动作的效果。比如人的行走动画，可以分为四个或十六个不同的图像。将这个图像存储在一个图像上，然后通过穷举其位置来渲染出流畅的动画。不过这个游戏因为资源与时间的限制，我们就没有了将这个技术付诸实践。

# 4 测试

吃下了一条小鱼以后，分数加一

 上图中玩家的鱼受伤后受到了惩罚性的扣分，同时大小以及级数也受到了减分

上图玩家吃到了珍珠而获得了一百分的奖励

上图玩家使用了清空小鱼的功能，分数减少10

根据测试，各项功能运行正常。

# 5 总结提高

## 5.1 课程设计总结

这次的课设我充分运用了VS2015，github等工具，我也严格的按照我自己的代码风格书写。总的下来，这些课设的代码量大概也在3k行左右了。也算个比较小的项目，但是大概超过了一个小程序的级别吧（大概）。我觉得收获挺多的。这个课设对我来说虽然不是说很难，但是我觉得既然是作业那就要严格按照作业要求来完成。所以我死心塌地得使用而easyX（其实是套了壳的GDI），也死心塌地地使用了C从头到脚实现了这个游戏。虽然很多思想都借鉴了OOPL的思想，以及C++的一些程序结构安排思路，但是还是严格的C程序。这样写反而倒是很有趣，就是可以避开一些可以大幅简化程序设计的函数。

我其实很享受这个过程，我很喜欢这个课设。因为这个课设是我大学以来做过自由度算是最大的一门课了。而且这门课我也完全能够玩得开哈哈，虽然是转专业的同学，我也并没有因为这个而去忽视或者敷衍这门课。因为我爱程序，我爱用机器的角度去思考。那种绝对逻辑绝对理性的处理方式，是我认为非常高效的。

很多人认为从头到尾的写未免会太过枯燥。但是我却认为这是个创造的过程。从一个一行都没有项目，逐渐增添成这么一个具有完整体系结构的系统，有它自己的处理方式，有它自己的消息传递方式。仿佛就像一个会思考的孩子，我将它从小带到大。

我从初中学习了第一门语言VB，我自己做了第一个程序，就是杀毒软件，就是简简单单使用CRC32以及MD5进行文件识别，用摘要黑名单做的一个简易的病毒库。当然，那个时候我怎么可能懂MD5，虽然得了个奖，但是看了网上网友的评论都说，这一定是抄的，都不是自己写的，我内心就特别的伤感，特别的委屈。我自己的代码，的确都是我自己写的，我查的资料，凭什么就说我是抄的呢？

后来高中我就自己学习C++，接触了OpenGL和DX，扬言自己要写一个游戏引擎，然后信誓旦旦地跟同学组了一个所谓的工作室。那个时候进展很慢，高中学业也很紧，我真的觉得自己很失败。

大学我落榜，被分配到了社会学系。我在大一想各种办法努力转专业，最终我成功了。我终于有了机会，能够自己安排时间写程序。也就是转专业的那个暑假，我用C++写了一个简单的框架，仅仅是一些零碎的组件。再过一个月，就是大二上学期了。

可以说是这个项目是我真的自己动手做的第一个成品。成就感可想而知。我现在看着它都觉得很兴奋。虽然写的还有很多不尽如人意的地方，不过我做到了，我以后会更加认真仔细地完成好自己的每一行代码。

感谢老师能够给我这个机会创造我自己喜欢的东西，现在转专业课程也很多，我也是挤出时间来写。事实上这个项目总共按照八小时工作日算的话，工时没有超过一周半。

## 5.2 对本课程意见与建议

我对这个课程特别的满意。尤其是我要感谢周珺老师。我能够特别开放的和周老师讨论技术实现的问题。而且周老师一直都很细心地帮助我解决具体问题。

我认为老师可以为同学提供一些在程序设计上的新思路，并且让同学能够通过自己的动手操作实现这些功能。并且同学在这之前并没有什么程序设计的经验，一般都是想到哪里就做到哪里。我也希望以后能够给后面的同学增加一些程序框架设计的课程来帮助他们构建更加合理清晰的程序结构。

目前大部分的同学都在使用VC6，而VC6早已经过时，这种将近二十年前的集成开发环境很明显是过时的，而同时大多同学以后需要使用更加先进的开发环境书写和测试代码。也许在测试环节会遇到很多的困难，那么这就需要他们自己探索测试的技巧，我认为这在这节课里面是十分缺乏的。同学们首先需要学习使用更加先进的开发环境，并熟悉各项测试技巧，才能够写出更加安全高效的代码。所以我希望以后的课程能够给同学介绍一些先进的开发环境，例如VS2015以及在其上的测试技巧等等。

## 5.3 附件：程序源代码

代码详见：https://github.com/mpskex/507-easyx