高级语言程序设计

课设报告

题　　目：Run Steve

学　　号：23072022

姓　　名：王 佳 伟

指导教师：杜 晓 琳

提交日期_____2024.05.22_____

成绩评价表

界面功能	程序执行情况	程序功能实现	问题回答情况	最终成绩
□内容丰富、炫酷 □扩展基本要求 □基本完成基本要求 □未完成基本要求	□流畅 □界面有闪动 □操作不灵活 □有错误	□模块和数据组织合理 □合理、少量不妥 □基本合理 □不合理	□清楚、正确 □基本正确 □回答有部分错误 □不能回答问题	
报告总体设计分析	**报告技术实现分析**	**其他内容及写作规范**	**总体评价**	
□深入、全面 □正确 □基本符合要求 □内容不全	□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	□清楚、正确 □基本正确 □回答有部分错误 □较混乱、不规范		

教师签字:_____杜晓林_____

目录

1 需求分析

参照游戏《地铁跑酷》，利用基础的 C 语言和 C++语言的 EasyX 图形库(包含在（<graphics.h>头文件中），同时借用著名游戏《我的世界》的游戏素材，制作一款简易的跑酷游戏。

1.1 功能需求

创建一个具有基本跑酷元素的游戏，包括角色（史蒂夫）自动奔跑、移动和跳跃躲避障碍物、挥剑攻击、收集金币等，同时保证游戏运行流畅，界面友好，适合各年龄段玩家。并额外增加登录注册系统，以及玩家积分排行榜。

1.1.1 角色控制

玩家能够控制角色（史蒂夫）左右移动、跳跃、挥剑攻击，以避开障碍物和拾取奖励。

1.1.2 障碍物与金币

随机生成不同类型的障碍物（如箭、僵尸等）和金币，增加游戏难度和趣味性。

1.1.3 得分系统

根据玩家收集的金币数量、击杀僵尸数量以及跑酷距离计算得分，设有高分排行榜。

1.1.4 道具系统

提供几种游戏内道具（如恢复血量、无敌加速等），增强游戏体验。

1.1.5 界面设计

包括开始界面、游戏界面、暂停界面、排行榜界面，具备清晰的操作提示和反馈。

1.1.6 音效与音乐

游戏应包含背景音乐和各种动作音效，给予玩家及时的操作反馈并提升沉浸感。

1.1.7 注册与登录系统

实现账户的创建于登录，存储玩家的游戏成绩。

1.1.8 排行榜

不同玩家在进行游戏后，会获得不同的分数，排行榜会根据用户获得的最高成绩由高到

低进行排行展示，并实时更新，激发玩家竞争意识和成就感。

1.2 数据需求

1.2.1 玩家账户信息链表

节点结构：每个节点包含玩家的用户名、密码，本局游戏积分，以及历史最高积分。链表依据玩家的历史最高积分有序排列，便于后续展示排行榜。

1.2.2 游戏障碍信息链表

节点结构：每个节点代表一个生成物实例，包括生成物类型（如箭、僵尸、金币等）、出现位置（第几个赛道以及 x 和 y 坐标）等。链表按游戏进程中的出现顺序排列，支持动态添加和移除。

1.3 界面需求

1.3.1 登录界面

游戏标题：醒目位置展示游戏名称“RUN-STEVE”，采用像素风字体和色彩吸引玩家注意，反映游戏的趣味性。

状态栏：位于游戏标题之下，实时显示登录状态（如“"Account not logged in”、“Login successful,Welcome 用户名”、“Login failed, Please try again”等）。

操作说明： 在状态栏下方简要展示游戏操作指南，如“↑ ↓ ← → CONTROL STEVE AND “SPACE” PAUSE”等，便于新玩家快速上手。

背景动画：通过逐帧循环播放游戏《我的世界》风景的视频，增加登录界面的美观性。同时确保视频播放流畅不干扰登录操作。

登录/注册按钮；点击后弹出登录/注册窗口包括“用户名”输入框和“密码”输入框，配以清晰的占位符文本指示输入内容。

开始游戏按钮：置于显著位置，点击后验证登录信息并进入游戏主界面。

1.3.2 游戏界面

血量显示：位于屏幕左上角，直观且不易遮挡游戏视野。采用心形图标形式展现，清晰反馈玩家当前生存状态。

背景天空随游戏进程昼夜变化：游戏开始时为清晨，随着游戏时间推移，天空逐渐过渡到白天、黄昏，最终进入夜晚，通过细腻的色彩渐变和星空、月亮、太阳等元素的动态展现，增强沉浸感。

积分显示：显示于屏幕右上角，使用白色，确保在任何背景下都清晰可见。积分数字动态增加，伴随特效（如上升动画）以庆祝得分。

游戏主体区域：占据屏幕中心，确保角色、障碍物、金币等关键游戏元素清晰可见，布局合理，避免拥挤感。角色设计生动，动作流畅，通过不同动作状态（奔跑、跳跃、挥剑）的精细动画表现，提升游戏的真实感和趣味性。障碍物随机但有序出现，难度随游戏进程逐

步增加，保证游戏的挑战性与平衡性。金币、道具等收集物分布沿途，鼓励玩家探索并采取策略性行动以最大化收益。

1.3.3 排行榜界面

Game Over 提示：位于界面中央上方，展示“GAME OVER”字样，确保玩家能立即意识到游戏结束状态。

玩家历史最高分展示：位于“GAME OVER”提示下方，以清晰的字体展示“您的历史最高分为：[具体分数]”。若当前玩家进入排行榜（只显示前十名），则会对当前玩家的信息做高亮处理，以提醒玩家。

玩家当前分数展示：在排行榜列表底部独立展示。

1.4 开发与运行环境需求

开发环境：Visual Studio 2022

运行环境：Windows11 系统

2. 游戏程序总体设计

2.1 程序模块设计

2.1.1 模块调用图

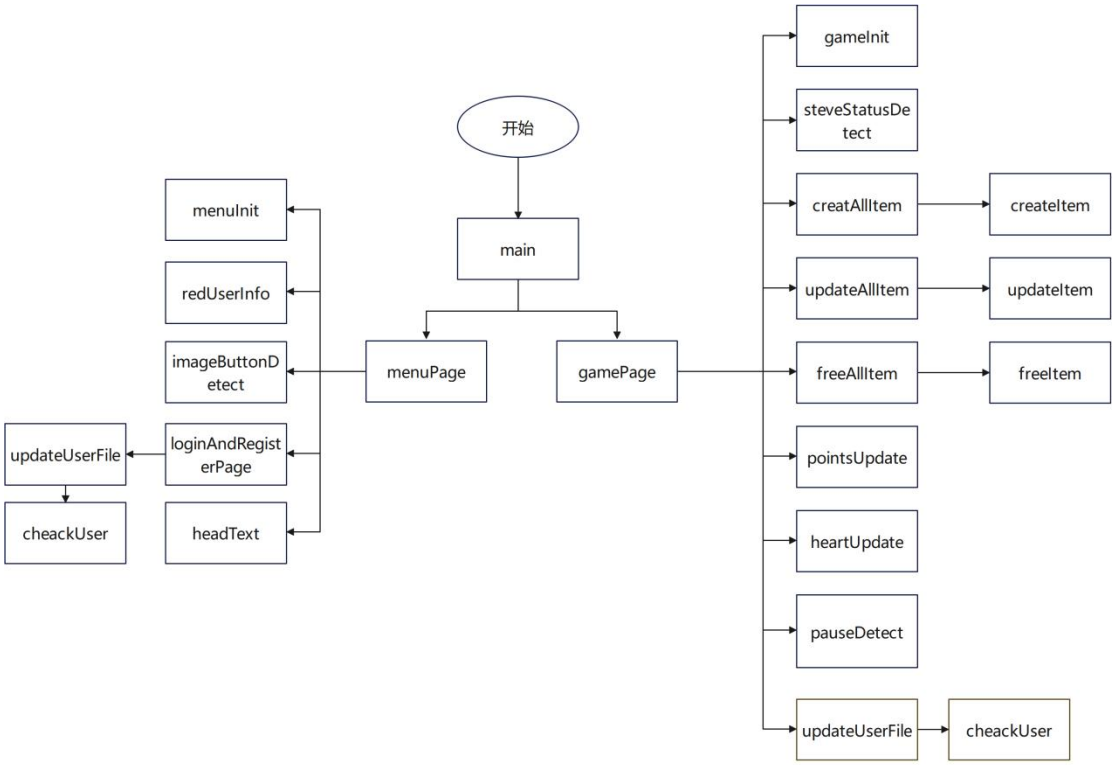


图 2-1 程序模块调用图

2.1.2 模块具体功能简述

模块编号：M0
函数原型：int main()
功能：程序主函数。
参数：无
返回值：0

模块编号：M0-1
函数原型：void menuPage()
功能：主界面循环。
参数：无
返回值：无

模块编号：M0-2
函数原型：void gamePage()
功能：游戏界面循环。
参数：无
返回值：无

模块编号：M1-1
函数原型：void menuInit()

功能：主界面初始化。
参数：无
返回值：无

模块编号：M1-2
函数原型：newUser* readUserInfo()
功能：将本地用户存档信息读取至链表。
参数：无
返回值：储存用户信息的链表头指针。

模块编号：M1-3
函数原型：bool imageButtonDetect(imageLocate& locate, IMAGE& image);
功能：检测鼠标点击位置是否在某个图像上。
参数：图片的位置 locate 以及图片信息 image。
返回值：鼠标点击位置在某个图片上返回真，否则返回假。

模块编号：M1-4
函数原型：void loginAndRegisterPage(IMAGE& page)
功能：登录和注册页面。
参数：登录或注册界面的图片 page。
返回值：无

模块编号：M1-4-1
函数原型：void updateUserFile()
功能：将游戏成绩信息存储至本地。
参数：无
返回值：无

模块编号：M1-4-2
函数原型：bool cheackUser()
功能：登录时检测输入的账户与本地存储的账户是否一致。
参数：无
返回值：输入的账户与本地存储的账户一致返回真 否则为假

模块编号：M1-5
函数原型：void headText()
功能：主界面显示登录状态。
参数：无
返回值：无

模块编号：M2-1
函数原型：void gameInit()
功能：游戏界面初始化。
参数：无

返回值：无

模块编号：M2-2
函数原型：void steveMove()
功能：史蒂夫移动/攻击状态检测。
参数：无
返回值：无

模块编号：M2-3
函数原型：void steveStatusDetect()
功能：史蒂夫移动/攻击/跳跃状态检测。
参数：无
返回值：无

模块编号：M2-4
函数原型：void creatAllItem()
功能：所有生成物的创建。
参数：无
返回值：无

模块编号：M2-4-1
函数原型：item* createItem(item* head,int modle ,int& cnt)
功能：创建该种生成物，并将其加入对应的生成物链表。
参数：该生成物链表头指针 head，新生成的生成物在第 modle 行，生成物的总个数 cnt
返回值：该种生成物链表的头指针。

模块编号：M2-5
函数原型：void updateAllItem()
功能：所有生成物的更新。
参数：无
返回值：无

模块编号：M2-5-1
函数原型：item* updateItem(item* headItem,int& cnt,int category)
功能：更新生成物链表 并将其显示出来。
参数：该种生成物链表的头指针，该类型生成物的个数，生成物的类型
返回值；该种生成物链表的头指针。

模块编号：M2-6
函数原型：void pointsUpdate()
功能：玩家分数更新。
参数：无
返回值：无

模块编号：M2-7
函数原型：void heartUpdate()
功能：玩家血量更新。
参数：无
返回值：无

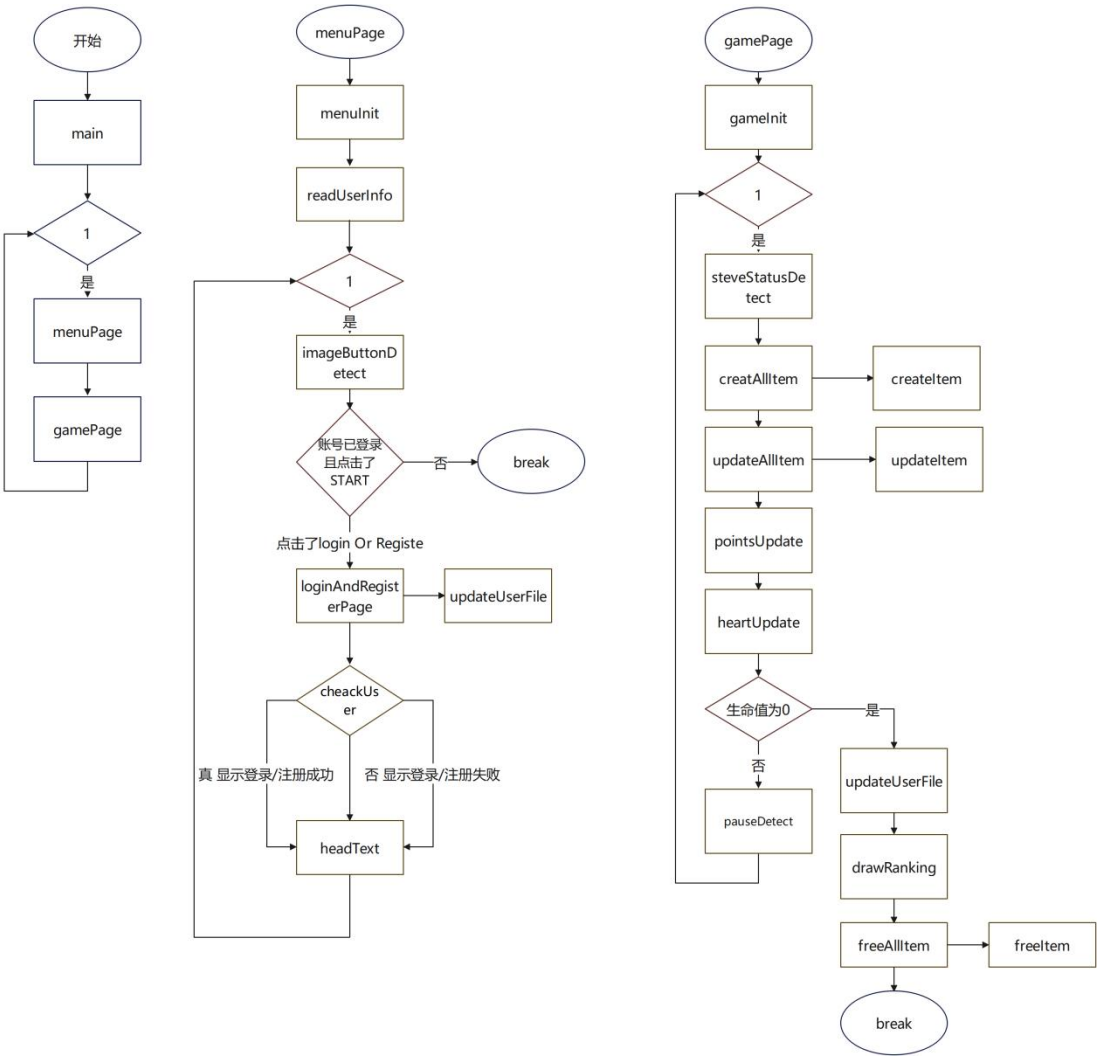
模块编号：M2-8
函数原型：void drawRanking()
功能：游戏排行榜绘制。
参数：无
返回值：无

模块编号：M2-9
函数原型：void freeAllItem()
功能：所有生成物的清理。
参数：无
返回值：无

模块编号：M2-9-1
函数原型：item* freeItem(item* barrierItem)
功能：清理该种生成物链表内存。
参数：该种生成物链表头指针
返回值：该种生成物链表头指针（NULL）。

模块编号：M2-
函数原型：void freeAllItem()
功能：所有生成物的清理。
参数：无
返回值：无

2.2 程序动态流程设计



2.3 主要数据结构

2.3.1 玩家账户信息结构

包括玩家的名称 name，密码 password，历史最高分 score，当前分数 points，指向下一个节点的指针 next。

```
struct newUser
{
    char name[21];
    char password[21];
    int score; //历史最高分
    int points; //当前分数
    newUser* next;
    newUser() : name("\0"),password("\0"),score(0),points(0),next(NULL) {}
```

```
};
```

2.3.2 游戏障碍信息链表

包括障碍的坐标(x,y)，障碍的速度 speed，障碍出现在第几列 modle，指向下一个节点的指针 next。

```
struct item
{
    int x;
    int y;
    int speed;
    int modle; //为出现在第几列  1 2 3
    item* next;
};
```

2.3.7 文件

位于./data/data.txt，用于按照玩家历史最高得分从大到小存储玩家信息，用于在登录时比对信息，并且同时用于展示玩家得分排行榜。

文件格式：

Name1
Password1
Score1
Name2
Password2
Score2
……

3 游戏实现技术点分析

3.1 静态画面设计

静态画面设计主要出现在主菜单界面和排行榜界面。其中菜单中的按钮和登录注册页面主要涉及图片的静态渲染，排行榜界面主要涉及文字的静态渲染。

图片的静态渲染依赖于 IMAGE 类和函数 loadimage 与 putimage。其中 IMAGE 类对象能够储存具有一定长宽的矩形图片，通过 loadimage 函数可以将本地图片加载至 IMAGE 对象中，在通过 putimage 函数可以将加载好的 IMAGE 对象图片绘制出来。

文字的绘制依赖于函数 outtextxy，能够实现文字的定点绘制。通过 settextcolor 和 settextstyle 可以设置文字的颜色和样式。

3.2 游戏动画设计

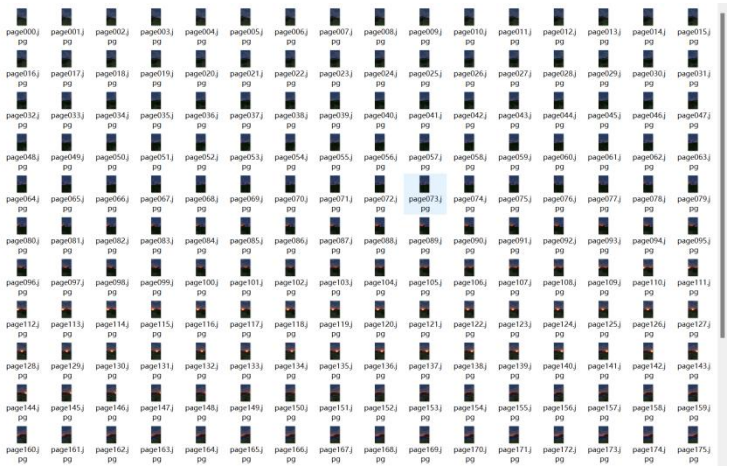
游戏动画设计在本程序中大量出现，如主菜单界面中的背景动画，游戏中主角 Steve 的移动、挥剑动作，游戏背景（道路、时间）的变化，游戏中生成物大小的不断变化形成的伪 3D 效果。

由于 EasyX 并不支持视频播放，所以所有的动画都只能变为逐帧图片播放。在制作素材中，使用了专业视频剪辑软件 Adobe Premiere Pro 将视频处理并以逐帧图片的形式保存。

为了使动画连贯流畅，在处理视频时特意注意将视频的第一帧与最后一帧重合或接近，时首位相接。

如图 3-1 主角 Steve 动画素材，共使用了 14 张逐帧图片。图 3-2 主页面背景动画素材，共使用了 240 张逐帧图片。这些图片首尾相接，使动画流畅连贯。

3-1 主角 Steve 动画素材



3-2 主页面背景动画素材

3.3 游戏交互设计

游戏交互部分包括在主界面玩家信息输入以及在游戏界面玩家对于主角 Steve 的控制。

玩家信息输入：在主界面玩家点击 LOGIN 或 REGISTE 按钮后，页面会显示出登录/注册页面。此时玩家可以先后输入用户名 Name 和密码 Password，按下 ENTER 确认，按下 BACKSPACE 回退。在输入过程中，系统会对输入的字数进行限制，以避免存储信息的数组越界。并且登录确认时程序会将玩家输入的用户名和密码和 data.txt 存储的已注册的玩家信息进行对比，并在主界面中显示登录状态（登录成功/登录失败/注册成功等）。

主角 Steve 控制：游戏过程中，玩家可以使用‘W’、‘A’、‘S’、‘D’（或‘↑’、‘↓’、‘←’、‘→’）来控制主角 Steve 分别进行跳跃、左移、挥剑、右移操作。按下‘SPACE’可暂停游戏。

3.4 其他技术点分析

3.4.1 透明图片显示

由于 EasyX 并不支持 png 格式的透明图片，故在绘制不规则图片中的透明部分时，需要准备两张图片，一张是黑底彩图，另一个为白底黑图的掩码图。例如图 3-3 所示。

gold.png　　gold1.png

图 3-3

这里主要使用位运算中的与运算和或运算。

先将白底黑图贴上进行与运算，其中白色的 RGB（255,255,255）二进制为 1111 1111，由于 1&1=1,1&0=0，故而白色与任何颜色进行与运算得到的还是该颜色，此时背景部分变为背景色。而黑色 RGB（0,0,0）二进制为 0000 0000，0&1=0,0&0=0，故而黑色与任何颜色进行与运算得到的都是黑色，此时图片部分仍是黑色。

之后使用黑底彩图进行或运算。由于 0|0=0,0|1=1，故而黑色与任何颜色进行或运算得到的还是该颜色，此时背景部分即为背景色。又因为此时图片部分是黑色，黑色与图片进行或运算得到的还是图片。此时就完成了透明图片的显示。

3.4.2 音乐播放

音乐的播放有两个函数可选，一个是 PlaySound，另一个是 mciSendString。通过这两个函数可以实现在 Windows 系统上调用多媒体进行打开，单词播放，循环播放，暂停播放等操作。

PlaySound 和 micSendString 都需要引用库<mmsystem.h>在 vs2010 以上版本需要加入 #pragma comment(lib, "winmm.lib")才能使用。

而这两个函数在同一时间均只能播放一个音乐，在播放音乐时，遇到下一个函数时会被打断，所以在程序中分别使用两个函数播放反馈音效和背景音乐。PlaySound 支持的格式仅为 wav，但播放效率快，故而用于播放反馈音效，使反馈更加及时。micSendString 对大部分媒体文件都适用，故而用于播放背景音乐。

例如：

```
PlaySound("../songs/button.wav", NULL, SND_ASYNC);

mciSendString("open ../songs/C418.wav alias MySong", NULL, 0, NULL);
mciSendString("play MySong", NULL, 0, NULL);
```

3.3.2 去除闪烁

在图片位移进行贴图的过程中，会造成图片留存的效应或是图片贴图闪烁、卡顿等问题。

为了解决这一问题，利用 BeginBatchDraw、 FlushBatchDraw 以及 EndBatchDraw 实现了双缓冲技术。

当 BeginBatchDraw 开始运行时，绘制的图形不会直接显示在屏幕上，而是存入到缓冲区里去，也就是存入到电脑的内存里面。当遇到 FlushBatchDraw 时存入的图片内容会释放出来，显示到屏幕上，就这样一直循环，实现每一次循环的每一帧图片可以显示到屏幕上，避

免画面撕裂。

```
……
BeginBatchDraw();
While(1)
{
    cleardevice();
    …… //绘制画面
    FlushBatchDraw()
}
EndBatchDraw();
```

4 测试

4.1 用户注册/登录输入测试

用户在注册或登录时只允许输入字母和数字，并且不能超过 14 位，超过后用户将无法继续输入。

5 用户手册

5.1“Run Steve”程序功能

本游戏是以跑酷类游戏“地铁跑酷”为参照，使用著名游戏“我的世界”为素材制作的一款跑酷类游戏。游戏提供账号注册登录系统，能够记录每位玩家的历史最高分数，并以排行榜的形式展示出来。在游戏中，玩家可以操控主角 Steve 进行移动、跳跃、挥剑动作来拾取金币和道具、躲避障碍、击杀僵尸等，并尝试获得更高的分数。

5.2“Run Steve”安装方法

将包含游戏和数据的压缩包解压缩，双击文件夹内的可执行文件（.exe）即可启动游戏。

5.3“Run Steve”游玩方法

打开 Run Steve.exe 文件后，可看到主界面，从上到下文字依次为标题、登录状态、游戏操作指南、开始游戏、登录、注册。如图 5-3-1。

图 5-3-1 主界面

需要登录账号后方可开始游戏。拥有账号的用户可点击 LOGIN 进行登录，没有账号的用户可点击 REGISTER 进行注册。在注册登录页面，玩家可以一次输入用户名和密码，每次输入完成后欧按“ENTER”确认。注册/登录界面如 5-3-2。

图 5-3-2 注册/登录界面

完成登录操作后，玩家可点击 START 进入游戏。在游戏界面，左上角为玩家当前的血量状态，右上角为玩家当前的分数。玩家可以使用‘W’、‘A’、‘S’、‘D’（或‘↑’、‘↓’、‘←’、‘→’）来控制主角 Steve 分别进行跳跃、左移、挥剑、右移操作，按下‘SPACE’可暂停游戏。如图 5-3-3

游戏过程中，将随机刷出以下道具：

金币，拾取获得 100 分数奖励；

箭，使用跳跃进行躲避，若撞上血量会减一；

僵尸，使用挥剑击杀，击杀后可获得 150 分数奖励，若撞上血量会减一；

无敌加速道具（金苹果），拾取后获得三秒的无敌加速效果；

回血道具，拾取后血量加一。

其中前三者的刷新概率会随游戏进行而增加。

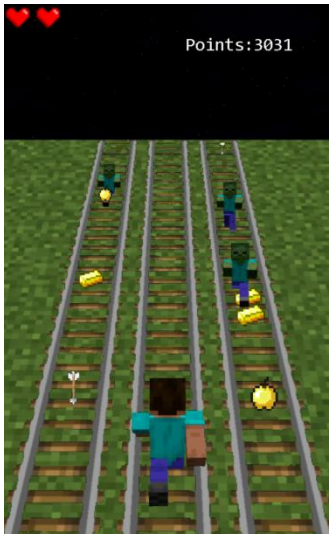


图 5-3-3 游戏界面

当玩家血量为 0 时，游戏结束，弹出排行榜界面。排行榜界面将显示出历史分数最高的前十名玩家，若玩家的历史最高分数进入前十名则会显示在排行榜中，并做高亮处理，以提醒玩家。下方将显示本次玩家的分数。按任意键便可以回到主界面。如图 5-3-4

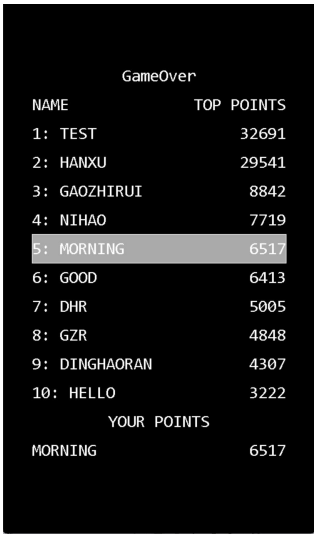


图 5-3-4

6 总结提高

6.1 课设设计总结

在这个学期的尾声，随着我的“Run Steve”课程设计的圆满落幕，我不仅亲手将课堂所学转化为实践成功，更在这一过程中经历了前所未有的成长与蜕变。

在课设开发的实践过程中，我深刻地领悟到了项目开发时所需要的前瞻规划与目标导向的重要性。即从项目伊始就要精心设计程序架构，明确数据处理的逻辑及其实现策略。起初，我缺乏这种先见之明，编码时随想随写，导致代码结构杂乱，维护困难，隐藏了不少缺陷。这使得后续的我不得不投入大量时间和精力进行代码的修正与重构，经历了一番波折。从此

以后，"预先规划，而后行码"的原则深深镌刻在我心中，成为我编程哲学的核心。

另外，在编写代码的过程中，不仅锻炼了我的编程技能，还学会了使用 git 这一强大的版本控制系统来高效地管理代码版本，确保了代码的可追溯性，为我的课设编写带来了极大的便利。

因此，随着课程的结束，我带走的不仅仅是完成的项目，更是一份对软件开发深刻的理解，一套实用的技能集，以及一颗勇于面对挑战、持续进步的心。我相信，这些宝贵的经验和教训将成为我未来学习生涯中的坚实基石，引导我在技术的征途中不断前行，探索无限可能。在此，我想向教授我 c 语言知识和为我 c 语言课设提供大量指导的杜晓林老师以及为我提出建议和帮助的同学致以最诚挚的感谢，是你们让我在这条充满挑战与机遇的路上，跑得更稳，也跑得更远。

6.2 对课程的一些建议

希望课程可以引入开源的思想，将历代同学们做的不同软件以及优秀代码发布在统一的平台上，既可以供后面的学弟学妹们学习，也可以使完成后的课设游戏发挥更大的余温。

在最后，再次由衷的感谢所有帮助过我的同学以及老师们！

7 附件：程序 源代码

```
#include<cstdio>
#include<graphics.h>
#include<conio.h>
#include<cstring>
#include<cstdlib>
#include<time.h>
#include<Windows.h>
#include <mmsystem.h>
#pragma comment(lib, "WINMM.LIB")

#define WIDTH 540
#define HEIGHT 920
FILE* dataFile; //  储存用户数据的文件
ExMessage msg;    //  存储用户操作信息
clock_t fpsMenu = 1000 / 165; //  主页每一帧的时间
clock_t fpsGame = 1000 / 60;    //  游戏页面每一帧的时间

//默认 SETTINGS 数值
const int AWARDPOINS = 100; //  奖励分数
const int HEARTCNT = 5; //生命数

// SETTINGS  随时间增加  部分属性会调整  即难度增大同时奖励积分增加
int award = AWARDPOINS; //  奖励分数基准
int heartCnt = HEARTCNT; //生命数
double goldRate = 0.001; //金币生成概率/每列每帧
double arrowRate = 0.001; //箭生成概率/每列每帧
double zombieRate = 0.001; //僵尸生成概率/每列每帧
double goldAppleRate = 0.0003; //无敌加速道具（金苹果）生成概率/每列每帧
double heartRate = 0.0003; //回血道具生成概率/每列每帧

//  用于稳定帧率
int startTime; //  某一帧的开始时间
int freamTime; //  加载某一帧所花费的时间

//  声明一些状态
int menuStatus; //0 为初始页面  1 为登录页面  2 为注册页面
int logORegStatus; //0 为初始  1 为输入账号状态  2 为输入密码状态  3 为输入完成状态
int userStatus; //0 为未登录  1 为登录成功  2 为登录失败  3 为注册成功
int steveModle = 2; // steve 当前所处的轨道  1 2 3（从左至右)
int jumpFlag = 0; // 1 jumping;
int attackFlag = 0; // 1 attacking
```

```
int invincibleFlag = 0; // 是否处于无敌状态
int invincibleStartTime; // 无敌时间
int hurtStatus=0; // 受伤状态 用于播放受伤反馈动画
int awardGoldStatus=0; // 拾取金币得分状态 用于播放拾取金币得分反馈动画
int awardZombieStatue=0; // 击杀僵尸的得分状态 用于播放击杀僵尸得分反馈动画
char Points[128]; // 存储将 int 分数转化为 char 字符串分数

int videoNum = 0; // 主界面视频播放第 x 帧数
int zombieImgCnt = 0; // 当前播放僵尸的第 x 帧
const int zombieNum = 14; // 僵尸动画帧数

// 主界面图片声明
IMAGE menu;
IMAGE login;
IMAGE register1;
IMAGE start;
IMAGE runSteve;
IMAGE menu_1;
IMAGE login_1;
IMAGE register_1;
IMAGE start_1;
IMAGE runSteve_1;
IMAGE lLogin;
IMAGE lRegister;
IMAGE lStart;
IMAGE lLogin_1;
IMAGE lRegister_1;
IMAGE lStart_1;
IMAGE loginPage;
IMAGE registerPage;
const int menuPageVideoNum = 240;
IMAGE menuPageVideoImage[menuPageVideoNum];


// 当前应该播放第 x 帧
int steveImageNum = 0;
int railImageNum = 0;
int backgroundImagNum = 0;
int skyImageNum = 0;
int attackImgCnt = 0;

// 动画总共帧数
const int steveNum = 14;
const int railNum = 4;
```

```
const int backgroundNum = 17;
const int skyNum = 24;
const int attackNum = 15;

// 图片声明
IMAGE gold[2];
IMAGE arrow[2];
IMAGE steve[steveNum];
IMAGE steve1[steveNum];
IMAGE attack[attackNum];
IMAGE attack1[attackNum];
IMAGE zomebie[2][zombieNum];
IMAGE rail[railNum];
IMAGE rail1[railNum];
IMAGE background[backgroundNum];
IMAGE sky[skyNum];
IMAGE heart[2];
IMAGE red;


// 游戏界面物品动画的逐帧图片
// 由于 loadimage 缩放图片效率过低
// 若在播放动画时加载 则会造成卡顿
// 故而先按比例加载好
// 以内存换时间 实测占用内存不多
// 较特殊 用全局变量更方便
IMAGE frameWiseGold[120];
IMAGE frameWiseGold1[120];
IMAGE frameWiseArrow[120];
IMAGE frameWiseArrow1[120];
IMAGE frameWiseZombie[120];
IMAGE frameWiseZombie1[120];
IMAGE frameWiseGoldApple[120];
IMAGE frameWiseGoldApple1[120];
IMAGE frameWiseHeart[120];
IMAGE frameWiseHeart1[120];


struct imageLocate
{
	int x;
	int y;
	imageLocate(int x, int y) : x(x), y(y) {}
};  // 用于储存静态（不需要放大的）图像坐标
```

```
//规定一些静态图片的位置
imageLocate runSteveLovate(12, 100);
imageLocate startLocate(132, 433);
imageLocate loginLocate(110, 580);
imageLocate registerLocate(21, 700);
imageLocate steveLocate(0, 0);
imageLocate railLocate(-90, -200);

struct item
{
	int x;
	int y;
	int speed;
	int modle; //为出现在第几列  1 2 3
	item* next;
}; //用链表来储存游戏生成物

struct newUser
{
	char name[21];
	char password[21];
	int score; //历史最高分
	int points; //当前分数
	newUser* next;
	newUser() : name("\0"),password("\0"),score(0),points(0),next(NULL) {}
}; //  链表来储存用户信息

newUser* head=NULL; //  储存用户信息的链表
newUser* user; //  当前登录的用户

//  当前生成物数量初始化
int goldCnt = 0;
int arrowCnt = 0;
int zombieCnt = 0;
int goldAppleCnt = 0;
int heartItemCnt = 0;

//  初始化生成物链表
item* itemGold = NULL;
item* itemArrow = NULL;
item* itemZombie = NULL;
item* itemGoldApple = NULL;
item* itemHeart = NULL;
```

```
void menuPage(); // 主界面循环
void menuInit(); // 主界面初始化
newUser* readUserInfo(); // 将本地用户存档信息读取至链表
bool imageButtonDetect(imageLocate& locate, IMAGE& image); // 检测鼠标点击位置是否在某个图像上
void loginAndRegisterPage(IMAGE& page);   // 登录和注册页面
void updateUserFile(); // 将游戏成绩信息存储至本地
bool cheackUser(); // 登录时检测输入的账户与本地存储的账户是否一致
void headText(); // 主界面显示登录状态
void gamePage(); // 游戏界面循环
void gameInit(); // 游戏界面初始化
void pauseDetect(); // 暂停检测
void steveStatusDetect(); // 史蒂夫移动/攻击状态检测
void creatAllItem(); // 所有生成物的创建
void updateAllItem(); // 所有生成物的更新
void freeAllItem(); // 所有生成物的清理
item* createItem(item* head,int modle ,int& cnt); // 创建该种生成物，并将其加入对应的生成物链表
item* updateItem(item* headItem,int& cnt,int category); // 更新生成物链表 并将其显示出来
item* freeItem(item* barrierItem); // 清理该种生成物链表内存
void heartUpdate(); // 玩家血量更新
void pointsUpdate(); // 玩家分数更新
void drawRanking(); // 游戏排行榜绘制


int main() //程序主函数
{
	initgraph(WIDTH, HEIGHT,1);
	while (true)
	{
		menuPage();
		gamePage();
	}
}

void menuPage()
{
	menuInit();
	head = readUserInfo();

	// 背景音乐播放
	mciSendString("open ./songs/C418.wav alias MySong", NULL, 0, NULL);
	mciSendString("play MySong", NULL, 0, NULL);
```

```
BeginBatchDraw();
while (true)
{
	startTime = clock();
	cleardevice();
	peekmessage(&msg);

	// 背景视频播放
	putimage(0, 0, &menuPageVideoImage[videoNum/2]);
	videoNum++;
	if (videoNum == 2*(menuPageVideoNum-1))
		videoNum = 0;

	if (imageButtonDetect(startLocate,start)) // 是否点击了 start
	{

		if (msg.message == WM_LBUTTONDOWN && (userStatus == 1 || userStatus == 3))
		{
			PlaySound("./songs/button.wav", NULL, SND_ASYNC);
			break;
		}

		putimage(startLocate.x, startLocate.y, &lStart_1, SRCAND);
		putimage(startLocate.x, startLocate.y, &lStart, SRCPAINT);
	}
	else
	{
		putimage(startLocate.x, startLocate.y, &start_1, SRCAND);
		putimage(startLocate.x, startLocate.y, &start, SRCPAINT);
	}
	if (imageButtonDetect(loginLocate, login)) // 是否点击了 login
	{
		if (msg.message == WM_LBUTTONDOWN)
		{
			menuStatus = 1;
			logORegStatus = 1;
			PlaySound("./songs/button.wav", NULL, SND_ASYNC);
		}
		else
		{
			putimage(loginLocate.x , loginLocate.y, &lLogin_1, SRCAND);
			putimage(loginLocate.x, loginLocate.y, &lLogin, SRCPAINT);
		}
```

```
}
else
{
	putimage(loginLocate.x, loginLocate.y, &login_1, SRCAND);
	putimage(loginLocate.x, loginLocate.y, &login, SRCPAINT);

}
if (imageButtonDetect(registerLocate, register1)) // 是否点击了 register
{
	if (msg.message == WM_LBUTTONDOWN)
	{
		logORegStatus = 1;
		menuStatus = 2;
		PlaySound("./songs/button.wav", NULL, SND_ASYNC);
	}
	else
	{
		putimage(registerLocate.x, registerLocate.y, &lRegister_1, SRCAND);
		putimage(registerLocate.x, registerLocate.y, &lRegister, SRCPAINT);
	}
}
else
{
	putimage(registerLocate.x, registerLocate.y, &register_1, SRCAND);
	putimage(registerLocate.x, registerLocate.y, &register1, SRCPAINT);
}
putimage(runSteveLovate.x, runSteveLovate.y, &runSteve_1, SRCAND);
putimage(runSteveLovate.x, runSteveLovate.y, &runSteve, SRCPAINT);

// 页面更新
if (menuStatus==1)
{

	loginAndRegisterPage(loginPage);
	if (logORegStatus ==3)
	{
		if (cheackUser())
			userStatus = 1;
		else
			userStatus = 2;
	}
}
else if (menuStatus == 2)
{
```

```
            loginAndRegisterPage(registerPage);
            if (logORegStatus == 3)
            {
                userStatus = 3;
            }
        }

        headText();
        FlushBatchDraw();

        // 帧率控制
        freamTime = clock() - startTime;
        if (fpsMenu-freamTime > 0)
            Sleep(fpsMenu - freamTime);

    }

    EndBatchDraw();
}

void menuInit()
{
    //用户信息初始化
    user = (newUser*)malloc(sizeof(newUser));
    user->name[0] = '\0';
    user->password[0] = '\0';
    user->points = 0;

    //状态初始化
    menuStatus = 0;
    logORegStatus = 0;
    userStatus = 0;
    steveModle = 2;
    jumpFlag = 0;
    attackFlag = 0;
    invincibleFlag = 0;
    hurtStatus = 0;
    char file_name[128];


    // 主界面图片加载
    for (int i = 0; i < menuPageVideoNum - 1; i++)
    {
        sprintf_s(file_name, "./image/menupagevideo/page%03d.jpg", i);
```

```
            //printf(file_name);
            loadimage(&menuPageVideoImage[i], file_name);
        }
        loadimage(&login, _T("./image/title/login.png"));
        loadimage(&register1, _T("./image/title/register.png"));
        loadimage(&start, _T("./image/title/start.png"));
        loadimage(&runSteve, _T("./image/title/Runsteve.png"));
        loadimage(&login_1, _T("./image/title/login_1.png"));
        loadimage(&register_1, _T("./image/title/register_1.png"));
        loadimage(&start_1, _T("./image/title/start_1.png"));
        loadimage(&runSteve_1, _T("./image/title/Runsteve_1.png"));
        loadimage(&lLogin, _T("./image/title/llogin.png"));
        loadimage(&lRegister, _T("./image/title/lregister.png"));
        loadimage(&lStart, _T("./image/title/lstart.png"));
        loadimage(&lLogin_1, _T("./image/title/llogin_1.png"));
        loadimage(&lRegister_1, _T("./image/title/lregister_1.png"));
        loadimage(&lStart_1, _T("./image/title/lstart_1.png"));
        loadimage(&loginPage, _T("./image/title/loginpage.png"));
        loadimage(&registerPage, _T("./image/title/registerpage.png"));

        settextstyle(35, 0, _T("Consolas"));
        settextcolor(BLACK);
        setbkcolor(WHITE);

}

bool imageButtonDetect(imageLocate& locate, IMAGE& image)
{
        if (msg.x >= locate.x && msg.x <= locate.x + image.getwidth() && msg.y >= locate.y && msg.y
<= locate.y + image.getheight())
            return 1;
        else
            return 0;
}

void loginAndRegisterPage(IMAGE& page)
{

        imageLocate loginPageLocate(44, 360);
        imageLocate userLocate(255, 415);
        imageLocate passwordLocate(255, 484);
        char ch[2];
        ch[1] = '\0';
```

```
if (logORegStatus==1)
{
	putimage(loginPageLocate.x, loginPageLocate.y, &page);
	if (msg.message == WM_KEYDOWN)
	{

		ch[0] = msg.vkcode;
		//printf("%d", ch[0]);
		if (ch[0] == 8 && strlen(user->name) > 0)
		{
			user->name[strlen(user->name) - 1] = '\0';
		}
		else if (ch[0] == 13)
		{
			logORegStatus = 2;
		}
		else
		{
			if (ch[0] != 8 && strlen(user->name) < 15&&((ch[0]>=48&&ch[0]<=57)||(ch[0]>=65&&ch[0]<=90)))
				strcat_s(user->name, ch);
		}
	}
	settextstyle(35, 15, _T("Consolas"));
	static int n = 0;
	if (n < 10)
	{
		n++;
		outtextxy(userLocate.x+15*strlen(user->name)-5, userLocate.y, "|");
	}
	else if(n>=10&&n<20)
	{
		n++;

	}
	else
	{
		n = 0;
	}
	outtextxy(userLocate.x, userLocate.y, user->name);
}
else if (logORegStatus==2)
{
	putimage(loginPageLocate.x, loginPageLocate.y, &page);
```

```
            if (msg.message == WM_KEYDOWN)
            {

                ch[0] = msg.vkcode;
                if (ch[0] == 8 && strlen(user->password) > 0)
                {
                    user->password[strlen(user->password) - 1] = '\0';
                }
                else if (ch[0] == 13)
                {
                    logORegStatus = 3;
                    if (menuStatus == 2)
                    {
                        userStatus = 3;
                    }
                    menuStatus = 0;
                }
                else
                {
                    if (ch[0] != 8 && strlen(user->password) < 15 && ((ch[0] >= 48 && ch[0] <= 57)
|| (ch[0] >= 65 && ch[0] <= 90)))
                        strcat_s(user->password, ch);
                }
            }
            settextstyle(35, 15, _T("Consolas"));
            static int n = 0;
            if (n < 10)
            {
                n++;
                outtextxy(passwordLocate.x + 15 * strlen(user->password) - 5, passwordLocate.y,
"|");
            }
            else if (n >= 10 && n < 20)
            {
                n++;

            }
            else
            {
                n = 0;
            }
            outtextxy(userLocate.x, userLocate.y, user->name);
            outtextxy(passwordLocate.x, passwordLocate.y, user->password);
        }
```

```
}

newUser* readUserInfo()
{
	head = NULL;
	fopen_s(&dataFile, "./data/data.txt", "r");
	while (!feof(dataFile))
	{
		newUser* p = (newUser*)malloc(sizeof(newUser));

		if (fgets(p->name, sizeof(p->name), dataFile) != NULL)    // 空文件检测
		{
		}
		else {
			break;
		}
		char temp[21];
		fgets(p->password, sizeof(p->password), dataFile);
		fgets(temp, sizeof(temp), dataFile);
		sscanf_s(temp, "%d", &p->score);
		p->name[strcspn(p->name,"\n")]='\0';
		p->password[strcspn(p->password,"\n")]= '\0';
		p->points = 0;
		//printf("%s\n%s\n", p->name, p->password);
		p->next = NULL;
		newUser* last = head;
		if (last)
		{
			while (last->next)
			{
				last = last->next;
			}
			last->next = p;
		}
		else
		{
			head = p;
		}
	}
	fclose(dataFile);
	return head;
}
```

```
void updateUserFile()
{
	if (!cheackUser())
	{
		//将 user 接入链表
		if (head == NULL)
		{
			head = user;
			user->next = NULL;
		}
		else
		{
			newUser* temp = head;
			while (temp->next!=NULL)
			{
				temp = temp->next;
			}
			temp->next = user;
			user->next = NULL;
		}
	}
	//将链表按 score 排序
	newUser* i, * j;
	char tempName[21];
	char tempPassword[21];
	int tempScore,tempPoints;
	char userName[21];
	strcpy_s(userName, user->name);
	for (i = head; i != NULL; i = i->next)
	{
		for (j = i; j != NULL; j = j->next)
		{
			if (j->score > i->score)
			{
				tempScore = i->score;
				tempPoints = i->points;
				strcpy_s(tempName, i->name);
				strcpy_s(tempPassword, i->password);
				i->score = j->score;
				i->points = j->points;
				strcpy_s(i->name, j->name);
				strcpy_s(i->password, j->password);
				j->score = tempScore;
				j->points = tempPoints;
```

```
                    strcpy_s(j->name, tempName);
                    strcpy_s(j->password, tempPassword);
                    if(!strcmp(i->name, userName))
                        user = i;
                }
            }
        }
        //将排好序的链表写入文件
        newUser* temp = head;
        fopen_s(&dataFile, "./data/data.txt", "w");
        while (temp != NULL)
        {
            fprintf(dataFile, "%s\n", temp->name);
            fprintf(dataFile, "%s\n", temp->password);
            fprintf(dataFile, "%d\n", temp->score);
            temp = temp->next;
        }
        fclose(dataFile);

}

bool cheackUser()
{
        newUser* temp = head;
        while (temp)
        {
            if (!strcmp(temp->name, user->name) && !strcmp(temp->password, user->password))
            {
                if(user->score>temp->score)
                    temp->score = user->score;
                user = temp;
                return 1;
            }
            temp = temp->next;
        }
        return 0;
}

void headText()
{
        setbkmode(TRANSPARENT);
        settextstyle(25, 0, _T("Consolas"));
        char s[50]="";
        char s2[50] = "Login failed, Please try again";
```

```
	char s3[50] = "";
	char s4[50] = "Account not logged in";
	char s5[60] = "↑ ↓ ← → CONTROL STEVE AND “SPACE” PAUSE";
	switch (userStatus)
	{
	case 0:
		outtextxy((WIDTH - textwidth(s4)) / 2, 200, s4);
		break;
	case 1:
		sprintf_s(s, "Login successful,Welcome %s", user->name);
		outtextxy((WIDTH-textwidth(s))/2, 200, s);
		break;
	case 2:
		outtextxy((WIDTH - textwidth(s2)) / 2, 200, s2);
		break;
	case 3:
		sprintf_s(s3, "Register successful,Welcome %s", user->name);
		outtextxy((WIDTH - textwidth(s3)) / 2, 200, s3);
		break;
	default:
		break;
	}
	outtextxy((WIDTH - textwidth(s5)) / 2, 230, s5);
}

//游戏循环
void gamePage()
{
	gameInit();
	BeginBatchDraw();

	while (true)
	{
		startTime = clock();
		cleardevice();
		peekmessage(&msg);


		//史蒂夫状态检测
		steveStatusDetect();

		//背景图片更新
		putimage(0, -50, &sky[skyImageNum / 100]);
		skyImageNum++;
```

```
if (skyImageNum == skyNum * 100)
	skyImageNum = 0;
putimage(0, 225, &background[backgroundImagNum]);
backgroundImagNum++;
if (backgroundImagNum == backgroundNum)
	backgroundImagNum = 0;
putimage(railLocate.x, railLocate.y, &rail1[railImageNum], SRCAND);
putimage(railLocate.x, railLocate.y, &rail[railImageNum], SRCPAINT);
railImageNum++;
if (railImageNum == railNum)
	railImageNum = 0;

// 生成物更新
creatAllItem();
updateAllItem();

// 史蒂夫动画更新
if (!attackFlag)
{
	putimage(steveLocate.x, steveLocate.y, &steve1[steveImageNum], SRCAND);
	putimage(steveLocate.x, steveLocate.y, &steve[steveImageNum], SRCPAINT);
	steveImageNum++;
	if (steveImageNum == steveNum)
		steveImageNum = 0;
}
else
{
	putimage(steveLocate.x, steveLocate.y, &attack1[attackImgCnt], SRCAND);
	putimage(steveLocate.x, steveLocate.y, &attack[attackImgCnt], SRCPAINT);
	attackImgCnt++;
	if (attackImgCnt == attackNum)
	{
		attackImgCnt = 0;
		attackFlag = 0;
	}
}

//受击动画更新
if (hurtStatus != 0)
{
	putimage(0, 0, &red, SRCAND);
	hurtStatus++;
	if (hurtStatus == 6)
		hurtStatus = 0;
```

```
            }

            pointsUpdate();
            heartUpdate();

            // 如果生命值为 0 执行游戏结束
            if (heartCnt == 0)
            {
                setbkcolor(BLACK);
                cleardevice();
                heartCnt = HEARTCNT;
                if (user->points > user->score)
                    user->score = user->points;
                updateUserFile();
                drawRanking();
                freeAllItem();
                break;
            }


            // 帧率控制 与 无敌加速（通过不限制帧率来实现加速效果）
            if (!invincibleFlag)
            {
                freamTime = clock() - startTime;
                if (fpsGame - freamTime > 0)
                    Sleep(fpsGame - freamTime);
            }
            else
            {
                if (clock() - invincibleStartTime > 3000)
                {
                    invincibleFlag = 0;
                }
            }
            pauseDetect();
            FlushBatchDraw();

        }
        EndBatchDraw();
}

void gameInit()
{
```

```
// 随位置变化的图片的加载 （伪 3D）
int size;
int y = 200;
char file_name[128];
char file_name1[128];
for (int i = 0; i < 120; i++)
{
	size = (int)((y * 0.1));
	loadimage(&frameWiseGold[i], "./image/star/gold1.png", size, size, true);
	loadimage(&frameWiseGold1[i], "./image/star/gold.png", size, size, true);
	loadimage(&frameWiseArrow[i], "./image/arrow/arrow1.png", size, size, true);
	loadimage(&frameWiseArrow1[i], "./image/arrow/arrow.png", size, size, true);
	loadimage(&frameWiseGoldApple[i], "./image/props/goldApple1.png", size, size, true);
	loadimage(&frameWiseGoldApple1[i], "./image/props/goldApple.png", size, size, true);
	loadimage(&frameWiseHeart[i], "./image/heart/heart1.png", size, size, true);
	loadimage(&frameWiseHeart1[i], "./image/heart/heart.png", size, size, true);
	sprintf_s(file_name, "./image/zombie/zombie%02d.jpg", zombieImgCnt);
	sprintf_s(file_name1, "./image/zombie1/zombie1%02d.jpg", zombieImgCnt);
	zombieImgCnt++;
	if (zombieImgCnt == zombieNum)
		zombieImgCnt = 0;
	loadimage(&frameWiseZombie[i], file_name1, size * 3, size * 3, true);
	loadimage(&frameWiseZombie1[i], file_name, size * 3, size * 3, true);
	y += 6;
}
loadimage(&red, "./image/red.png");
loadimage(&gold[0], "./image/star/gold1.png");
loadimage(&gold[1], "./image/star/gold.png");
loadimage(&arrow[0], "./image/arrow/arrow1.png");
loadimage(&arrow[1], "./image/arrow/arrow.png");
loadimage(&heart[0], "./image/heart/heart.png", 50, 50, true);
loadimage(&heart[1], "./image/heart/heart1.png", 50, 50, true);
for (steveImageNum = 0; steveImageNum < steveNum; steveImageNum++)
{
	sprintf_s(file_name, "./image/steve/steve%02d.jpg", steveImageNum);
	sprintf_s(file_name1, "./image/steve1/steve101%02d.jpg", steveImageNum);
	//printf(file_name);
	loadimage(&steve[steveImageNum], file_name);
	loadimage(&steve1[steveImageNum], file_name1);

}
for (railImageNum = 0; railImageNum < railNum; railImageNum++)
{
```

```
        sprintf_s(file_name, "./image/rail/rail%d.jpg", railImageNum);
        sprintf_s(file_name1, "./image/rail1/rail1%d.jpg", railImageNum);
        loadimage(&rail[railImageNum], file_name, 720, 1280, true);
        loadimage(&rail1[railImageNum], file_name1, 720, 1280, true);
    }
    for (backgroundImagNum = 0; backgroundImagNum < backgroundNum; backgroundImagNum++)
    {
        sprintf_s(file_name, "./image/background/background%02d.jpg", backgroundImagNum);
        loadimage(&background[backgroundImagNum], file_name);
    }
    for (skyImageNum = 0; skyImageNum < skyNum; skyImageNum++)
    {
        sprintf_s(file_name, "./image/sky/sky%02d.jpg", skyImageNum);
        loadimage(&sky[skyImageNum], file_name);
    }
    for (attackImgCnt = 0; attackImgCnt < attackNum; attackImgCnt++)
    {
        sprintf_s(file_name, "./image/attack/attack%02d.jpg", attackImgCnt);
        sprintf_s(file_name1, "./image/attack1/attack1%02d.jpg", attackImgCnt);
        //printf(file_name);
        loadimage(&attack[attackImgCnt], file_name);
        loadimage(&attack1[attackImgCnt], file_name1);
    }
    for (zombieImgCnt = 0; zombieImgCnt < zombieNum; zombieImgCnt++)
    {
        sprintf_s(file_name, "./image/zombie/zombie%02d.jpg", zombieImgCnt);
        sprintf_s(file_name1, "./image/zombie1/zombie1%02d.jpg", zombieImgCnt);
        //printf(file_name);
        loadimage(&zomebie[0][zombieImgCnt], file_name);
        loadimage(&zomebie[1][zombieImgCnt], file_name1);
    }

    // 当前播放第 x 帧 重置
    attackImgCnt = 0;
    zombieImgCnt = 0;
    steveImageNum = 0;
    railImageNum = 0;
    backgroundImagNum = 0;
    skyImageNum = 1100;

    setbkmode(TRANSPARENT);
    settextcolor(WHITE);
```

```
}

void pauseDetect()
{
	if (msg.message == WM_KEYDOWN && msg.vkcode == 32)
	{
		setbkmode(TRANSPARENT);
		settextstyle(25, 0, _T("Consolas"));
		char s[10] = "PAUSE";
		outtextxy((WIDTH - textwidth(s)) / 2, (HEIGHT - textheight(s)) / 2, s);
		FlushBatchDraw();
		while (true)
		{
			peekmessage(&msg);
			if (msg.message == WM_KEYDOWN && msg.vkcode == 32)
				break;
		}
	}
}

void steveStatusDetect()
{
	static int	steveMoveFlag = 0; //-1 为向左移动  1 为向右移动  0 为不动
	int steveSpeed = 10; //速度应可以整除 150
	const int steveLeft = -180;
	const int steveRight = 180;
	if (msg.message == WM_KEYDOWN)
	{
		if (msg.vkcode == 37||msg.vkcode== 65)
		{
			steveMoveFlag = -1;
		}
		if (msg.vkcode == 39||msg.vkcode== 68)
		{
			steveMoveFlag = 1;
		}
		if (msg.vkcode == 0x28||msg.vkcode== 83)
		{
			attackFlag = 1;
		}
	}

	if (steveMoveFlag == -1 && steveLocate.x > steveLeft)
```

```
            steveLocate.x -= steveSpeed;
        else if (steveMoveFlag == 1 && steveLocate.x < steveRight)
            steveLocate.x += steveSpeed;
        if (steveLocate.x == steveLeft || steveLocate.x == steveRight || steveLocate.x == 0)
        {
            steveMoveFlag = 0;
            if (steveLocate.x == steveLeft)
                steveModle = 1;
            if (steveLocate.x == steveRight)
                steveModle = 3;
            if (steveLocate.x == 0)
                steveModle = 2;
        }

        static int v0 = 26;
        static int gravity = 2;

        if (msg.message == WM_KEYDOWN)
        {
            if (msg.vkcode == 38 || msg.vkcode == 87)
                jumpFlag = 1;
        }
        if (jumpFlag == 1)
        {
            steveLocate.y -= v0;
            v0 -= gravity;
        }

        if (steveLocate.y == 0)
        {
            v0 = 26;
            jumpFlag = 0;
        }

}

void creatAllItem()
{
        //srand((unsigned int)time(0));
        int rand_number = rand() % 100000 + 1;

        //难度和奖励会随分数增加而增加
        goldRate = 0.001 + user->points * 0.000001;
        arrowRate = 0.001 + user->points * 0.000001;
```

```
	zombieRate = 0.001 + user->points * 0.000001;

	if (rand_number < goldRate * 3 * 100000)
	{
		rand_number = rand_number % 3 + 1;
		itemGold = createItem(itemGold, rand_number, goldCnt);
	}
	else if (rand_number < arrowRate * 3 * 100000 + goldRate * 3 * 100000)
	{
		rand_number = rand_number % 3 + 1;
		itemArrow = createItem(itemArrow, rand_number, arrowCnt);
	}
	else if (rand_number < zombieRate * 3 * 100000 + arrowRate * 3 * 100000 + goldRate * 3 *
100000)
	{
		rand_number = rand_number % 3 + 1;
		itemZombie = createItem(itemZombie, rand_number, zombieCnt);
	}
	else if (goldAppleCnt <= 1 && rand_number < goldAppleRate * 3 * 100000 + zombieRate * 3
* 100000 + arrowRate * 3 * 100000 + goldRate * 3 * 100000)
	{
		rand_number = rand_number % 3 + 1;
		itemGoldApple = createItem(itemGoldApple, rand_number, goldAppleCnt);
	}
	else if (heartItemCnt <= 1 && rand_number < heartRate * 3 * 100000 + goldAppleRate * 3 *
100000 + zombieRate * 3 * 100000 + arrowRate * 3 * 100000 + goldRate * 3 * 100000)
	{
		rand_number = rand_number % 3 + 1;
		itemHeart = createItem(itemHeart, rand_number, heartItemCnt);
	}
}

void updateAllItem()
{
	itemGold = updateItem(itemGold, goldCnt, 1);
	itemZombie = updateItem(itemZombie, zombieCnt, 3);
	itemArrow = updateItem(itemArrow, arrowCnt, 2);
	itemGoldApple = updateItem(itemGoldApple, goldAppleCnt, 4);
	itemHeart = updateItem(itemHeart, heartItemCnt, 5);
}

void freeAllItem() {
	itemGold = freeItem(itemGold);
	itemZombie = freeItem(itemZombie);
```

```
	itemArrow = freeItem(itemArrow);
	itemGoldApple = freeItem(itemGoldApple);
	itemHeart = freeItem(itemHeart);
}

item* createItem(item* head,int modle,int& cnt)
{
	cnt++;
	item* p = new item();
	p->next = NULL;
	p->speed = 6;
	p->y = 200;
	p->modle = modle;

	if (modle == 1)
		p->x = 170;
	else if (modle == 2)
		p->x = 250;
	else if (modle == 3)
		p->x = 340;

	item* last = head;
	if (last)
	{
		while (last->next)
		{
			last = last->next;
		}
		last->next = p;
	}
	else
	{
		head = p;
	}
	return head;
}

item* updateItem(item* headItem,int& cnt,int category) //category 1 为金币  2 为箭头  3 为僵
尸
{
	IMAGE image[2];
	if (headItem!=NULL && headItem->y > HEIGHT)
	{
		item* toDelete = headItem;
```

```
            headItem = headItem->next;
            free(toDelete);
            cnt--;
        }
        if (category==3&&headItem != NULL && headItem->y >= 600 && headItem->y <= 750 &&
headItem->modle == steveModle && attackFlag == 1)
        {
            item* toDelete = headItem;
            headItem = headItem->next;
            free(toDelete);
            cnt--;
            awardZombieStatue = 1;
            user->points += award * 2;
            PlaySound("./songs/zombieDied.wav", NULL, SND_ASYNC);
        }
        if (headItem != NULL && headItem->y >= 750 && headItem->y <= 800 && headItem->modle
== steveModle&&jumpFlag==0)
        {
            item* toDelete = headItem;
            headItem = headItem->next;
            free(toDelete);
            cnt--;
            if (category == 1)
            {
                user->points += award;
                awardGoldStatus = 1;
                PlaySound("./songs/getGold.wav", NULL, SND_ASYNC);
            }
            if ((category == 2||category==3)&&!invincibleFlag)
            {
                heartCnt--;
                hurtStatus = 1;
                PlaySound("./songs/hurt.wav", NULL, SND_ASYNC);
            }
            if (category == 4&& !invincibleFlag)
            {
                invincibleStartTime = clock();
                invincibleFlag = 1;
                PlaySound("./songs/goldApple.wav", NULL, SND_ASYNC);
            }
            if (category == 5)
            {
                PlaySound("./songs/getGold.wav", NULL, SND_ASYNC);
                heartCnt++;
```

```
        }
    }
    item* head = headItem;
    int size;
    while (head)
    {
        if (category ==3&&head->next != NULL && head->next->modle == steveModle &&
attackFlag == 1)
        {
            if (head->next->y >= 600 && head->next->y <= 750)
            {
                item* toDelete = head->next;
                head->next = head->next->next;
                free(toDelete);
                cnt--;
                awardZombieStatue = 3;
                user->points += award * 2;
                PlaySound("./songs/zombieDied.wav", NULL, SND_ASYNC);
            }
        }
        if (head->next != NULL && head->next->modle == steveModle&&jumpFlag == 0)
        {
            if (head->next->y >= 750 && head->next->y <= 800)
            {
                item* toDelete = head->next;
                head->next = head->next->next;
                free(toDelete);
                cnt--;
                if (category == 1)
                {
                    user->points += award;
                    awardGoldStatus = 1;
                    PlaySound("./songs/getGold.wav", NULL, SND_ASYNC);
                }

                if ((category == 2 || category == 3) && !invincibleFlag)
                {
                    heartCnt--;
                    hurtStatus = 1;
                    PlaySound("./songs/hurt.wav", NULL, SND_ASYNC);
                }
                if (category == 4&&!invincibleFlag)
                {
                    invincibleStartTime = clock();
```

```
                invincibleFlag = 1;
                PlaySound("./songs/goldApple.wav", NULL, SND_ASYNC);
            }
            if (category == 5)
            {
                PlaySound("./songs/getGold.wav", NULL, SND_ASYNC);
                heartCnt++;
            }
        }
}

head->y += head->y*0.015;
//printf("%d\n", head -> speed);

if (head->modle == 1)
{
    head->x = (int)(-0.224 * head->y +220);
}
else if (head->modle == 2)
{
    head->x = (int)(-0.05 * head->y+270);
}
else if (head->modle == 3)
{
    head->x = (int)(0.13*head->y+320);
}


size = (int)((head->y * 0.1));
int imageCnt = (head->y - 200) / 6;
if (category == 1)
{
    putimage(head->x, head->y, &frameWiseGold[imageCnt], SRCAND);
    putimage(head->x, head->y, &frameWiseGold1[imageCnt], SRCPAINT);
}
else if (category == 2)
{

    putimage(head->x, head->y, &frameWiseArrow[imageCnt], SRCAND);
    putimage(head->x, head->y, &frameWiseArrow1[imageCnt], SRCPAINT);
}
else if (category == 3)
{
```

```
                putimage(head->x-size, head->y, &frameWiseZombie[imageCnt], SRCAND);
                putimage(head->x -size, head->y, &frameWiseZombie1[imageCnt], SRCPAINT);
            }
            else if (category == 4)
            {
                putimage(head->x , head->y, &frameWiseGoldApple[imageCnt], SRCAND);
                putimage(head->x , head->y, &frameWiseGoldApple1[imageCnt], SRCPAINT);
            }
            else if(category==5)
            {
                putimage(head->x, head->y, &frameWiseHeart[imageCnt], SRCAND);
                putimage(head->x, head->y, &frameWiseHeart1[imageCnt], SRCPAINT);
            }
            head = head->next;
        }
        return headItem;
}

item* freeItem(item* barrierItem)
{
        while (barrierItem!=NULL)
        {
            item* toFree = barrierItem;
            barrierItem = barrierItem->next;
            free(toFree);
        }
        return barrierItem;
}

void heartUpdate()
{
        imageLocate heartLocate(0, 0);
        for (int i = 0; i < heartCnt;i++)
        {
            putimage(heartLocate.x, heartLocate.y, &heart[1], SRCAND);
            putimage(heartLocate.x, heartLocate.y, &heart[0], SRCPAINT);
            heartLocate.x += 50;
        }
}

void pointsUpdate()
{
        static int y = 90;
        char addPoints[20];
```

```
	if (awardGoldStatus == 1|| awardZombieStatue==1)
	{
		y = 90;
	}
	sprintf_s(Points, "Points:%04d", user->points);
	if (awardGoldStatus != 0)
		sprintf_s(addPoints, "+%d", award);
	else if (awardZombieStatue!=0)
		sprintf_s(addPoints, "+%d", award*2);
	settextstyle(35, 0, _T("Consolas"));
	outtextxy(300, 50, _T(Points));
	if (awardGoldStatus != 0)
	{
		outtextxy(410, y, addPoints);
		awardGoldStatus++;
		y--;
	}
	if (awardZombieStatue != 0)
	{
		outtextxy(410, y, addPoints);
		awardZombieStatue++;
		y--;
	}
	if (awardGoldStatus == 20)
	{
		awardGoldStatus=0;
	}
	if (awardZombieStatue == 20)
	{
		awardZombieStatue = 0;
	}
	user->points++;
}

void drawRanking()
{
	const int recRight = 50;
	const int recLeft = WIDTH - 50;
	const int recHigh = 50;
	const int recTop = 100;
	RECT r[15];
	newUser* temp = head;
	for (int i = 0; i < 15; i++)
	{
```

```
            r[i] = {recRight,(recTop + recHigh * i),recLeft,(recTop + recHigh * (i + 1))};
            if (i == 0)
            {
                drawtext(_T("GameOver"), &r[i], DT_CENTER | DT_VCENTER | DT_SINGLELINE);
            }
            else if (i == 1)
            {
                drawtext(_T("TOP POINTS"), &r[i], DT_RIGHT | DT_VCENTER | DT_SINGLELINE);
                drawtext(_T("NAME"), &r[i], DT_LEFT | DT_VCENTER | DT_SINGLELINE);
            }
            else if(i>1&&i<12)
            {
                if (temp == NULL)
                    continue;
                if (!strcmp(temp->name, user->name))
                {
                    setfillcolor(LIGHTGRAY);
                    fillrectangle(recRight, (recTop + recHigh * i), recLeft, (recTop + recHigh * (i +
1)));
                }
                char score[21] = "";
                char name[21] = "";
                sprintf_s(score, "%d", temp->score);
                sprintf_s(name, "%d: %s", i-1, temp->name);
                drawtext(score, &r[i], DT_RIGHT | DT_VCENTER | DT_SINGLELINE);
                drawtext(name, &r[i], DT_LEFT | DT_VCENTER | DT_SINGLELINE);
                temp = temp->next;
            }
            else if (i == 12)
            {
                drawtext(_T("YOUR POINTS"), &r[i], DT_CENTER | DT_VCENTER | DT_SINGLELINE);
            }
            else if (i == 13)
            {
                char score[21] = "";
                char name[21] = "";
                sprintf_s(score, "%d", user->points);
                sprintf_s(name, "%s", user->name);
                drawtext(score, &r[i], DT_RIGHT | DT_VCENTER | DT_SINGLELINE);
                drawtext(name, &r[i], DT_LEFT | DT_VCENTER | DT_SINGLELINE);
            }
        }
        FlushBatchDraw();
        while (1) {
```

```
            peekmessage(&msg);
            if (msg.message == WM_KEYDOWN)
                  break;
      }
}
```