# 

**高级语言程序设计**

**课设报告**

**题 目\_\_\_\_\_\_\_\_\_\_\_采蘑菇\_\_\_\_\_\_\_\_\_\_\_**

**学 号\_\_\_\_\_\_\_\_\_15071018\_\_\_\_\_\_\_\_\_\_\_**

**姓 名\_\_\_\_\_\_\_\_\_\_\_刘云\_\_\_\_\_\_\_\_\_\_\_\_\_**

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

**提交日期\_\_\_\_\_\_\_\_\_2016.4.29\_\_\_\_\_\_\_\_\_\_**

**成绩评价表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **报告内容** | **报告结构** | **报告图表** | **报告与程序一致性** | **最终成绩** |
| **□丰富正确**  **□完整正确**  **□基本正确**  **□问题很大** | **□完全符合要求**  **□基本符合要求**  **□有比较多的缺陷**  **□完全不符合要求** | **□符合规范**  **□基本符合规范**  **□有一些错误**  **□完全不正确** | **□完全一致**  **□基本一致**  **□基本不一致** |  |
| **程序功能实现** | **程序执行情况** | **问题回答情况** | **总体评价** | |
| **□完成基本功能和扩展功能**  **□完成基本功能几乎无扩展功能**  **□基本完成基本功能**  **□未完成基本功能** | **□顺畅**  **□有问题，经过老师指出之后改正**  **□有问题，无法改正** | **□立即正确回答**  **□经思考后正确回答**  **□回答有部分错误**  **□回答完全错误**  **□不能回答问题** |  | |

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

目录

[1 需求分析 3](#_Toc449556144)

[1.1 功能需求 3](#_Toc449556145)

[1.1.1 基本功能 3](#_Toc449556146)

[1.1.2 附加功能 3](#_Toc449556147)

[1.2 数据需求 3](#_Toc449556148)

[1.3 界面需求 4](#_Toc449556149)

[1.4 开发与运行环境需求 4](#_Toc449556150)

[1.5 其他方面需求 4](#_Toc449556151)

[2 概要设计 5](#_Toc449556152)

[2.1 主要数据结构 5](#_Toc449556153)

[2.1.1 GrassNode 5](#_Toc449556154)

[2.1.2 Game 5](#_Toc449556155)

[2.1.3 Player 6](#_Toc449556156)

[2.1.4 存档文件MRS 6](#_Toc449556157)

[2.2 程序总体结构 7](#_Toc449556158)

[2.2.1 模块调用图 7](#_Toc449556159)

[2.2.2 主程序流程图 8](#_Toc449556160)

[2.3 子模块设计 8](#_Toc449556161)

[2.3.1 main.cpp 8](#_Toc449556162)

[2.3.2 game\_main.cpp 8](#_Toc449556163)

[2.3.3 game\_menu.cpp 9](#_Toc449556164)

[2.3.4 winapi.cpp 9](#_Toc449556165)

[2.3.5 game.cpp 9](#_Toc449556166)

[2.3.6 player.cpp 10](#_Toc449556167)

[3 详细设计 11](#_Toc449556168)

[4 测试 13](#_Toc449556169)

[4.1 键盘鼠标测试 13](#_Toc449556174)

[4.2 文件测试 14](#_Toc449556175)

[4.3 排行榜测试 14](#_Toc449556176)

[5 用户手册 14](#_Toc449556177)

[6 总结提高 14](#_Toc449556178)

[6.1 课程设计总结 14](#_Toc449556179)

[6.2 对本课程意见与建议 15](#_Toc449556180)

[6.3 附件：程序源代码 15](#_Toc449556181)

# 需求分析

用 C 语言编写一个简单的“采蘑菇”游戏，采到蘑菇得分，采到炸弹减分，分数是随机的，范围在1到9之间。

## 功能需求

### 基本功能

开始游戏前，输入用户名和游戏的总时间，时间到游戏停止。初始时界面上显示一些草，位置随机，草上面有个分数，分是一位整数（随机产生，范围在1-9之间），草的下面可能是蘑菇，可能是炸弹，也可能什么都没有。点击开始按钮，人从游戏区下面中间的位置出发，向上走动，人走动的速度是匀速的，若想改变方向通过按上下左右键完成，若没有按键，走到边界自动改变方向，时间到游戏停止。草、蘑菇和炸弹的位置随机产生。走到草的位置时按下空格键则显示出草下面是什么，碰到蘑菇得分，碰到炸弹2秒内若能离开，不减分，炸弹仍然显示，然后蘑菇或炸弹消失；每隔10秒再增加一个草（蘑菇或炸弹或空）；若走到草的位置，没按空格键，则草不变化。游戏开始后，在信息显示区动态显示本盘游戏玩家的得分。点击清空按钮，则把所有的蘑菇、炸弹，草都清除，用户名和分数不变；点击“退出”按钮，把当前用户名和得分追加到指定文件中（即文件中存储了所有玩过此游戏的用户名和分数），释放空间，退出程序。

### 附加功能

将草、蘑菇、炸弹、人的当前状态，用户名、分数和剩余时间存入文件。将文件中存储的游戏状态恢复到屏幕上，继续游戏。

设定草的初始个数，每次增加的个数，间隔时间。

使游戏更加有趣的功能，或使界面更加美观的功能（例如，鼠标单击控制区中的按钮时，按钮有被按下去的效果，松开时，有弹回来的效果）。

## 数据需求

输入数据：用户名，游戏时间，初始个数，每次增加个数，间隔时间。导入已保存的游戏数据。

中间数据：蘑菇的位置、分值等动态变化的信息、游戏进行的状态（暂停、继续）、剩余时间。

输出数据：草、蘑菇、炸弹的动态显示、游戏的导出、玩家用户名、分数的记录。

## 界面需求

界面分为四个区域：游戏区、控制区和信息显示区。游戏界面如图所示。







**5**

**1**

用户名：aa

得分：10

剩余时间

20

得分：10

图1.1 游戏界面示例

程序运行时先输入用户名和游戏时间，再显示游戏界面， 信息显示区显示游戏的状态信息，例如用户名、所得分数，剩余时间；时间到游戏停止。

单击“开始”按钮，游戏开始。

单击“清空”按钮，把所有的蘑菇、炸弹，草都清除，游戏区只有一个人。

单击“退出”按钮，把当前用户名和得分追加到指定文件中，即文件中存储了所有玩过此游戏的用户名和分数。释放空间，结束程序。

扩充功能的按钮，可放在控制区，例如存状态，导入，结束，继续等功能。

## 开发与运行环境需求

开发工具：Visual Studio 2015

环境需求：Windows 10

## 其他方面需求

程序健壮性良好，考虑到用户各种可能的操作行为，避免出现错误。

# 概要设计

模块化设计。主要模块有：

1. 绘制主菜单
2. 绘制主界面、草、蘑菇、炸弹、玩家
3. 游戏状态控制，暂停和继续
4. 按钮点击、键盘操作
5. 玩家的移动和改变方向
6. 采蘑菇操作
7. 草的生成和删除，炸弹爆炸
8. 计算玩家分数
9. 游戏状态导出、导入
10. 记录用户名和分数
11. 退出并保存相关信息

## 主要数据结构

### GrassNode

struct GrassNode {

int id; 标识符

int type; 类型：蘑菇/炸弹/空

int grass\_style; 草的样式

int score; 分数

int x; 坐标 X

int y; 坐标 Y

bool picked; 是否已采蘑菇

bool exploded; 是否已爆炸

time\_t time\_picked; 采蘑菇的时间

GrassNode \*next; 下一节点

static int grid[3][4]; 网格中是否有草

};

### Game

struct Game {

wchar\_t player\_name[11]; 玩家姓名

int time\_left; 剩余时间

int score; 分数

int init\_num; 初始个数

int num\_at\_a\_time; 每次增加个数

int interval; 间隔时间

int grass\_num; 当前草个数

int last\_id; 最后的草的标识符

GrassNode \*h; 头节点

GrassNode \*grass\_focus; 当前范围内草节点

int button\_focus; 按钮焦点

bool button\_on\_click; 按钮按下

bool paused; 游戏暂停状态

bool grayscale\_ready; 画好了灰度图像

bool on\_exit; 点击了退出按钮

};

### Player

struct Player {

int skin; 皮肤

int x; 坐标 X

int y; 坐标 Y

int dx; X 轴方向速度

int dy; Y 轴方向速度

int speed; 移动速度

int direction; 移动方向

};

### 存档文件MRS

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 第1行 | 玩家姓名 | 剩余时间 | 分数 | 当前草个数 | 每次增加个数 | 间隔时间 | 最后草的标识符 |
| 第2行 | 玩家皮肤 | 坐标 X | 坐标 Y | X 轴方向速度 | Y 轴方向速度 | 移动速度 | 移动方向 |
| 第3行 ~EOF | 标识符 | 类型 | 分数 | 坐标 X | 坐标 Y | 是否已采蘑菇 | 是否已爆炸 |

## 程序总体结构

### 模块调用图

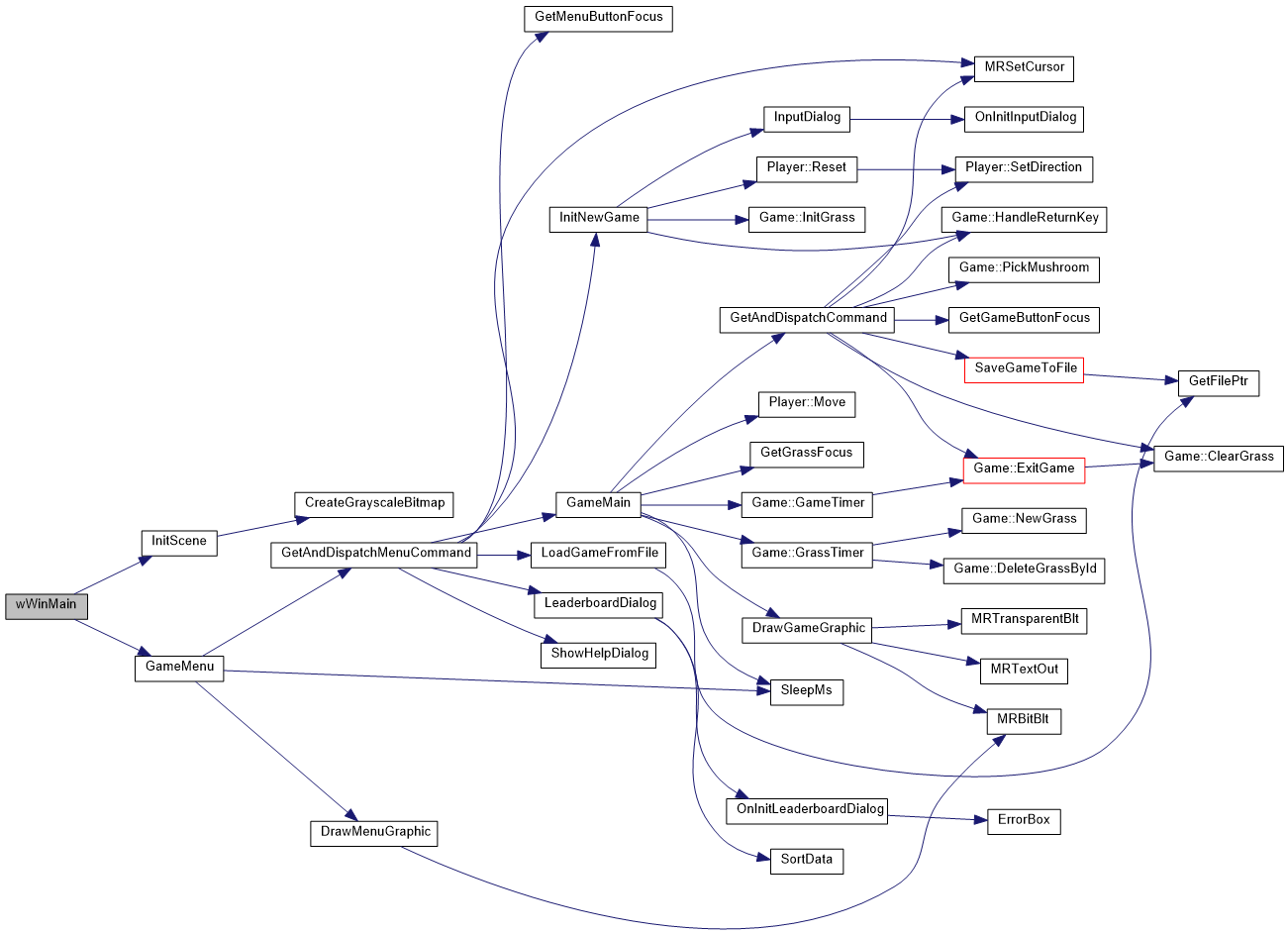


图2.1 模块调用图

### 主程序流程图

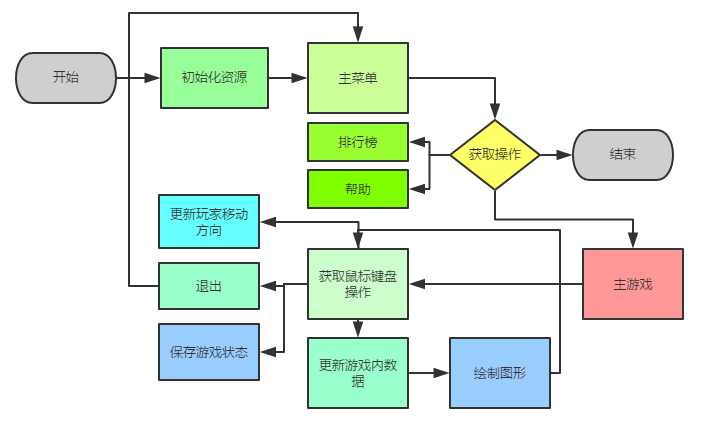


图2.2 主程序流程图

## 子模块设计

### main.cpp

void InitGraphics(IMAGE \*images, HDC hdc[]);

功能：初始化，加载图片

void SaveGameToFile(Game &game, Player &player);

功能：保存游戏到存档文件

bool LoadGameFromFile(Game &game, Player &player);

功能：从存档文件加载游戏

### game\_main.cpp

void GameMain(Game &game, Player &player, HDC hdc[]);

功能：游戏主循环

bool InitNewGame(Game &game, Player &player);

功能：初始化新游戏

void DrawGameGraphic(HDC hdc[], Game &game, Player &player);

功能：绘制游戏图形

void GetAndDispatchCommand(Game &game, Player &player);

功能：获取键盘和鼠标命令并执行操作

void GetGrassFocus(Game &game, Player &player);

功能：获取当前的草焦点

int GetGameButtonFocus(int x, int y);

功能：获取按钮焦点

### game\_menu.cpp

void GameMenu(Game &game, Player &player, HDC hdc[]);

功能：菜单主循环

void DrawMenuGraphic(HDC hdc[], Game &game);

功能：绘制菜单图形

void GetAndDispatchMenuCommand(Game &game, Player &player, HDC hdc[]);

功能：获取键盘和鼠标命令并执行操作

int GetMenuButtonFocus(int x, int y);

功能：获取按钮焦点

### winapi.cpp

int CALLBACK InputDialog(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

功能：“新游戏”对话框回调函数

int CALLBACK LeaderboardDialog(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

功能：“排行榜”对话框回调函数

void SortData(wchar\_t data[50][3][11], int n, int item, bool descending);

功能：排行榜数据排序

int ShowExitGameDialog(int score, bool timeout);

功能：显示“退出游戏”对话框

void ShowHelpDialog();

功能：显示“帮助”对话框

wfstream GetFileStream(int mode, wchar\_t\* filename);

功能：获取读/写文件流

### game.cpp

void InitGrass();

功能：初始化新游戏时创建一定数量的草

void NewGrass();

功能：创建一个新草

void DeleteGrassById(int id);

功能：删除指定标识符的草

void ClearGrass();

功能：清除所有草

void PickMushroom();

功能：在当前位置采蘑菇

void GameTimer();

功能：游戏剩余时间计时器

void GrassTimer();

功能：新建草、删除草、炸弹计时器

void SaveScoreToLeaderboard();

功能：保存分数至排行榜

void ExitGame(bool timeout);

功能：处理“退出游戏”对话框结果

void HandleReturnKey();

功能：控制回车键的按键频率，防止反复暂停/继续

void Reset();

功能：重置分数、游戏状态

### player.cpp

void Move();

功能：移动玩家位置

void SetDirection(Direction d);

功能：设置玩家移动方向

void Reset();

功能：重置玩家位置和方向

# 详细设计



图3.1 主菜单

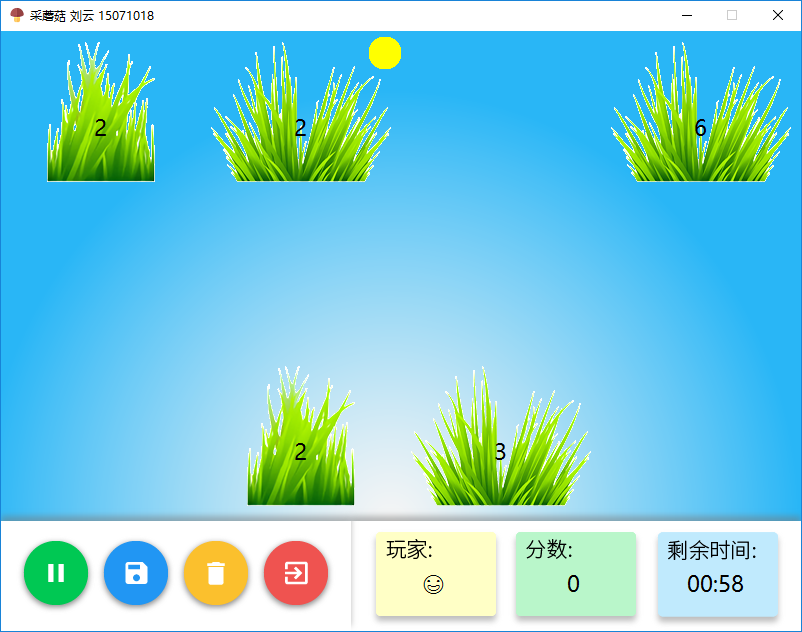


图3.2 游戏界面

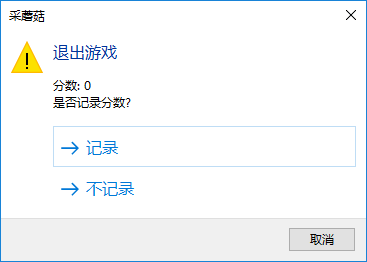
 

图3.3 新游戏界面 图3.4 退出游戏提示

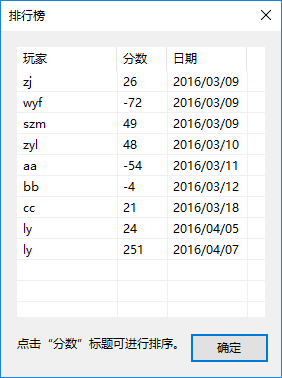
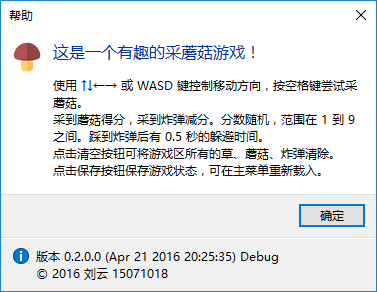
 

图3.5 排行榜界面 图3.6 帮助界面

* 1. **主循环**

图3.7 主循环流程图

* 1. **绘制图形**

图3.8 绘制图形流程图

* 1. **获取键盘鼠标事件**

图3.9 获取键盘鼠标事件流程图

* 1. **更新游戏数据**

图3.10 更新游戏数据流程图

# 测试



## 键盘鼠标测试

测试目的：检测游戏是否按照用户的键盘操作执行。

测试方法：开始后通过点击“开始新游戏”，输入用户名并确定。进入游戏，通过键盘的上下左右键或WASD键，测试玩家是否随按键移动。按下空格键采蘑菇。用鼠标点击暂停、清空、保存、退出按钮。

## 文件测试

测试目的：检测程序的文件处理功能。

测试方法：在游戏界面保存当前游戏状态，退出游戏。主菜单载入保存的游戏。载入早期版本程序生成的存档文件时可能导致程序错误。

## 排行榜测试

测试目的：检测程序的排行榜功能。

测试方法：在主菜单选择“排行榜”，打开排行榜。点击“分数”进行排序。需要一个有效的“leaderboard.txt”，无效的文件将不显示任何分数。

# 用户手册

1. 应用程序功能的详细说明；

一个简单的“采蘑菇”游戏，采到蘑菇得分，采到炸弹减分，分数是随机的，范围在1到9之间。

(2) 应用程序运行环境的要求；

Windows 10, Microsoft Visual C++ 2015 Redistributable。

(3) 应用程序的安装与启动方法；

不需要安装。直接运行Release目录下的mushroom.exe。

(4) 程序的界面、交互方式和操作方法；

使用鼠标和键盘完成操作。

(5) 输入数据类型、格式和内容限制；

用户名不超过10个字符。其他输入数据在允许的范围内。

(6) 在应用程序运行中，用户需要使用的交互命令名称、功能和格式的详细说明。

不需要交互命令。使用键盘和鼠标即可。

# 总结提高

## 课程设计总结

C语言课程只介绍了语言的基本语法，并没有关于制作大型项目的提示。在编写程序的过程中，我从毫无头绪，到逐步测试想法，程序逐渐成形，到最后进行修正错误和优化，我学习到了许多重要的知识和技能。使用大量子函数使结构更加清晰。为了实现图形效果，查阅了EasyX和Windows GDI的文档，了解到许多函数，从而实现了暂停时界面变灰色的效果。学习wchar\_t的使用，研究Unicode和其他编码的区别，从而解决了乱码的问题。EasyX的InputBox函数效果简陋，于是学习MSDN上的关于Windows API的文档，实现了新游戏的图形界面。EasyX绘制的图形并不精细，于是用Adobe Illustrator绘制了背景、按钮等图片。随后出现了图片太多，加载麻烦的问题，用TexturePacker将小图片打包成一张大图片。经过制作课程设计的考验，我对编程工具、图片处理工具都有了更多的了解。在测试的过程中练习了断点、添加监视等调试技巧，写程序更加熟练了。

## 对本课程意见与建议

建议使用较新的图形库。EasyX缺少基本的PNG格式支持，功能也比较少。虽然可以使用其他技巧实现透明贴图，但图片边缘有锯齿，效果不好。

## 附件：程序源代码

#pragma once

#pragma comment(lib, "comctl32.lib")

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

#include <Windows.h>

#include <commctrl.h>

#include <Shlwapi.h>

#include <time.h>

#include <math.h>

#include <fstream>

#include <codecvt>

#include "lib/graphics.h"

#include "resource.h"

using namespace std;

#ifdef \_DEBUG

const wchar\_t kBuildType[] = L"Debug";

#else

const wchar\_t kBuildType[] = L"Release";

#endif

const wchar\_t kVersion[] = L"0.2.0.0";

const wchar\_t kMushroom[] = L"采蘑菇";

const int kWidth = 800;

const int kHeight = 600;

const int kBottom = 490;

enum Grass { MUSHROOM, BOMB, NOTHING };

enum Direction { LEFT, UP, RIGHT, DOWN };

struct GrassNode {

int id;

int type;

int style;

int score;

int x;

int y;

bool picked;

bool exploded;

time\_t time\_picked;

GrassNode \*next;

static int grid[3][4];

GrassNode(int i);

};

struct Game {

wchar\_t player\_name[11];

int time\_left;

int score;

int init\_num;

int num\_at\_a\_time;

int interval;

int grass\_num;

int last\_id;

GrassNode \*h;

GrassNode \*grass\_focus;

int button\_focus;

bool button\_on\_click;

bool paused;

bool grayscale\_ready;

bool on\_exit;

Game();

void InitGrass();

void NewGrass();

void DeleteGrassById(int id);

void ClearGrass();

void PickMushroom();

void GameTimer();

void GrassTimer();

void SaveScoreToLeaderboard();

void ExitGame(bool timeout);

void HandleReturnKey();

void Reset();

};

struct Player {

const int kPlayerSize[2] = { 16,40 };

int skin;

int x;

int y;

int dx;

int dy;

int speed;

int direction;

Player();

void Move();

void SetDirection(Direction d);

void Reset();

};

inline int MRBitBlt(HDC hdc, const int xy[], HDC hdcsrc, const int src[]) {

return BitBlt(hdc, xy[0], xy[1], src[2], src[3], hdcsrc, src[0], src[1], SRCCOPY);

}

inline int MRTransparentBlt(HDC hdc, int x, int y, HDC hdcsrc, int w, int h, int xsrc, int ysrc) {

return GdiTransparentBlt(hdc, x - w / 2, y - h / 2, w, h, hdcsrc, xsrc, ysrc, w, h, 0x0);

}

inline int MRTransparentBlt(HDC hdc, int x, int y, HDC hdcsrc, const int src[]) {

return GdiTransparentBlt(hdc, x - src[2] / 2, y - src[3] / 2, src[2], src[3], hdcsrc, src[0], src[1], src[2], src[3], 0x0);

}

inline int MRTextOut(HDC hdc, const int t[], wchar\_t str[]) {

return ExtTextOut(hdc, t[0], t[1], 0, nullptr, str, wcslen(str), nullptr);

}

inline int MRTextOut(HDC hdc, int x, int y, wchar\_t str[]) {

return ExtTextOut(hdc, x, y, 0, nullptr, str, wcslen(str), nullptr);

}

inline int ErrorBox(const wchar\_t str[]) {

return TaskDialog(GetHWnd(), GetModuleHandle(nullptr), kMushroom, nullptr, str, TDCBF\_OK\_BUTTON, TD\_ERROR\_ICON, nullptr);

}

inline int InfoBox(const wchar\_t str[]) {

return TaskDialog(GetHWnd(), GetModuleHandle(nullptr), kMushroom, nullptr, str, TDCBF\_OK\_BUTTON, TD\_INFORMATION\_ICON, nullptr);

}

int CALLBACK InputDialog(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

int CALLBACK LeaderboardDialog(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

void SortData(wchar\_t data[50][3][11], int n, int item, bool descending);

bool OnInitInputDialog(HWND hWnd);

bool OnInitLeaderboardDialog(HWND hDlg, wchar\_t data[50][3][11]);

int ShowExitGameDialog(int score, bool timeout);

void ShowHelpDialog();

wfstream GetFileStream(int mode, wchar\_t\* filename);

void CreateGrayscaleBitmap(HDC hdc);

void MRSetCursor(int focus);

extern wchar\_t temp\_name[11]; extern int temp\_num[6];

void InitGraphics(IMAGE \*images, HDC hdc[]);

void SaveGameToFile(Game &game, Player &player);

bool LoadGameFromFile(Game &game, Player &player);

void SleepMs(int ms);

void GameMain(Game &game, Player &player, HDC hdc[]);

bool InitNewGame(Game &game, Player &player);

void DrawGameGraphic(HDC hdc[], Game &game, Player &player);

void GetAndDispatchCommand(Game &game, Player &player);

void GetGrassFocus(Game &game, Player &player);

int GetGameButtonFocus(int x, int y);

void GameMenu(Game &game, Player &player, HDC hdc[]);

void DrawMenuGraphic(HDC hdc[], Game &game);

void GetAndDispatchMenuCommand(Game &game, Player &player, HDC hdc[]);

int GetMenuButtonFocus(int x, int y);

int WINAPI wWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, PWSTR pCmdLine, int nCmdShow) {

IMAGE images[3];

HDC hdc[5];

Game game;

Player player;

InitGraphics(images, hdc);

GameMenu(game, player, hdc);

return 0;

}

void InitGraphics(IMAGE \*images, HDC hdc[]) {

initgraph(kWidth, kHeight);

SetWindowText(GetHWnd(), L"采蘑菇 刘云 15071018");

HICON icon = LoadIcon(GetModuleHandle(nullptr), MAKEINTRESOURCE(IDI\_MUSHROOM));

SendMessage(GetHWnd(), WM\_SETICON, ICON\_BIG, (LPARAM)icon);

DestroyIcon(icon);

LOGFONT f;

gettextstyle(&f);

f.lfHeight = 30;

wcscpy\_s(f.lfFaceName, L"微软雅黑");

f.lfQuality = CLEARTYPE\_QUALITY;

settextstyle(&f);

loadimage(images + 0, L"IMAGE", MAKEINTRESOURCE(IDR\_IMAGERES));

loadimage(images + 1, L"IMAGE", MAKEINTRESOURCE(IDR\_UI));

loadimage(images + 2, L"IMAGE", MAKEINTRESOURCE(IDR\_MENU));

hdc[0] = GetImageHDC();

hdc[1] = GetImageHDC(images);

hdc[2] = GetImageHDC(images + 1);

hdc[3] = GetImageHDC(images + 2);

hdc[4] = CreateCompatibleDC(hdc[0]);

CreateGrayscaleBitmap(hdc[4]);

SetTextColor(hdc[0], BLACK);

SetBkMode(hdc[0], TRANSPARENT);

SetTextAlign(hdc[0], TA\_CENTER);

}

void SaveGameToFile(Game &game, Player &player) {

wchar\_t filename[30];

swprintf\_s(filename, L"%s.mrs", game.player\_name);

wchar\_t t = '\t';

wfstream f = GetFileStream(1, filename);

if (!f.is\_open())

return;

f << game.player\_name << t << game.time\_left << t << game.score << t << game.grass\_num << t << game.num\_at\_a\_time << t << game.interval << t << game.last\_id << endl;

f << player.skin << t << player.x << t << player.y << t << player.dx << t << player.dy << t << player.speed << t << player.direction << endl;

GrassNode \*p = game.h->next;

for (int i = 0; i < game.grass\_num; i++) {

f << p->id << t << p->type << t << p->style << t << p->score << t << p->x << t << p->y << t << p->picked << t << p->exploded << endl;

p = p->next;

}

f.close();

wchar\_t buffer[40];

swprintf\_s(buffer, L"游戏已保存至 %s", filename);

InfoBox(buffer);

}

bool LoadGameFromFile(Game &game, Player &player) {

wfstream f = GetFileStream(0, nullptr);

if (!f.is\_open())

return false;

game.Reset();

f >> game.player\_name >> game.time\_left >> game.score >> game.grass\_num >> game.num\_at\_a\_time >> game.interval >> game.last\_id;

f >> player.skin >> player.x >> player.y >> player.dx >> player.dy >> player.speed >> player.direction;

game.h = new GrassNode(-1);

GrassNode \*p = game.h, \*s;

for (int i = 0; i < game.grass\_num; i++) {

s = new GrassNode(i);

f >> p->id >> p->type >> p->style >> p->score >> p->x >> p->y >> p->picked >> p->exploded;

GrassNode::grid[p->y][p->x] = 1;

p->next = s;

p = p->next;

}

f.close();

return true;

}

void SleepMs(int ms) {

static clock\_t old = clock();

old += ms;

if (clock() > old) {

old = clock();

return;

}

while (clock() < old)

Sleep(1);

}

void GameMenu(Game &game, Player &player, HDC hdc[]) {

while (true) {

GetAndDispatchMenuCommand(game, player, hdc);

DrawMenuGraphic(hdc, game);

SleepMs(5);

}

}

void DrawMenuGraphic(HDC hdc[], Game &game) {

const int kMenuItemsXY[5][2] = { { 430,180 },{ 430,250 },{ 430,320 },{ 430,390 },{ 430,460 } };

const int kMenuItems[10][4] = { { 252,124,250,40 },{ 1,124,250,40 },{ 252,165,250,40 },{ 252,83,250,40 },{ 133,1,250,40 },{ 384,1,250,40 },{ 252,42,250,40 },{ 1,83,250,40 },{ 1,42,250,40 },{ 1,165,250,40 } };

BeginBatchDraw();

BitBlt(hdc[0], 0, 0, kWidth, kHeight, hdc[3], 0, 0, SRCCOPY);

if (game.button\_focus != -1 && !game.button\_on\_click)

MRBitBlt(hdc[0], kMenuItemsXY[game.button\_focus], hdc[1], kMenuItems[game.button\_focus]);

if (game.button\_on\_click)

MRBitBlt(hdc[0], kMenuItemsXY[game.button\_focus], hdc[1], kMenuItems[5 + game.button\_focus]);

EndBatchDraw();

}

void GetAndDispatchMenuCommand(Game &game, Player &player, HDC hdc[]) {

MOUSEMSG message;

if (MouseHit()) {

message = GetMouseMsg();

switch (message.uMsg) {

case WM\_MOUSEMOVE:

game.button\_focus = GetMenuButtonFocus(message.x, message.y);

if (game.button\_focus == -1)

game.button\_on\_click = false;

MRSetCursor(game.button\_focus);

break;

case WM\_LBUTTONDOWN:

game.button\_on\_click = game.button\_focus != -1 ? true : false;

break;

case WM\_LBUTTONUP:

game.button\_on\_click = false;

switch (game.button\_focus) {

case 0:

if (InitNewGame(game, player))

GameMain(game, player, hdc);

break;

case 1:

if (LoadGameFromFile(game, player))

GameMain(game, player, hdc);

break;

case 2:

DialogBox(GetModuleHandle(nullptr), MAKEINTRESOURCE(IDD\_LEADERBOARDDIALOG), GetHWnd(), LeaderboardDialog);

break;

case 3:

ShowHelpDialog();

break;

case 4:

closegraph();

exit(0);

}

break;

}

}

}

int GetMenuButtonFocus(int x, int y) {

const int kButtons[5][2] = { { 555,200 },{ 555,270 },{ 555,340 },{ 555,410 },{ 555,480 } };

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

if (x > kButtons[i][0] - 125 && y > kButtons[i][1] - 20 && x < kButtons[i][0] + 125 && y < kButtons[i][1] + 20)

return i;

}

return -1;

}

void GameMain(Game &game, Player &player, HDC hdc[]) {

while (!game.on\_exit) {

GetAndDispatchCommand(game, player);

if (!game.paused) {

player.Move();

GetGrassFocus(game, player);

game.GameTimer();

game.GrassTimer();

}

DrawGameGraphic(hdc, game, player);

SleepMs(5);

}

}

wchar\_t temp\_name[11]; int temp\_num[6];

bool InitNewGame(Game &game, Player &player) {

if (DialogBox(GetModuleHandle(nullptr), MAKEINTRESOURCE(IDD\_INPUTDIALOG), GetHWnd(), InputDialog) == IDCANCEL)

return false;

srand(clock());

wchar\_t emoji[3] = { (wchar\_t)55357,(wchar\_t)(56832 + rand() % 80),'\0' };

wcscpy\_s(game.player\_name, temp\_name[0] ? temp\_name : emoji);

game.time\_left = temp\_num[0];

game.init\_num = temp\_num[1];

game.num\_at\_a\_time = temp\_num[2];

game.interval = temp\_num[3];

player.speed = temp\_num[4];

player.skin = temp\_num[5];

player.Reset();

game.HandleReturnKey();

game.Reset();

game.InitGrass();

return true;

}

void DrawGameGraphic(HDC hdc[], Game &game, Player &player) {

const int kTextsXY[3][2] = { { 433,537 },{ 573,537 },{ 715,537 } };

const int kButtonsXY[4][2] = { { 14,505 },{ 94,505 },{ 174,505 },{ 254,505 } };

const int kButtons[6][4] = { { 487,246,80,80 },{ 1,327,80,80 },{ 406,246,80,80 },{ 325,246,80,80 },{ 244,246,80,80 },{ 568,246,80,80 } };

const int kPlayer[8][4] = { { 34,1,32,32 },{ 100,1,32,32 },{ 1,1,32,32 },{ 67,1,32,32 },{ 565,165,67,80 },{ 503,165,61,80 },{ 95,246,67,80 },{ 163,246,80,80 } };

const int kMushrooms[9][4] = { { 329,327,64,90 },{ 1,418,65,90 },{ 591,327,43,90 },{ 172,327,77,86 },{ 82,327,89,83 },{ 496,327,94,90 },{ 1,246,93,80 },{ 394,327,101,90 },{ 250,327,78,90 } };

const int kGrass[3][4] = { { 1,559,187,140 },{ 325,418,108,140 },{ 434,418,180,140 } };

const int kGrassHighlight[3][4] = { { 189,559,187,140 },{ 558,559,108,140 },{ 377,559,180,140 } };

const int kBomb[2][4] = { { 67,418,145,100 },{ 213,418,111,103 } };

const int kGrassX[4] = { 100,300,500,700 };

const int kGrassY[3] = { 81,243,405 };

wchar\_t buffer[11];

BeginBatchDraw();

BitBlt(hdc[0], 0, 0, kWidth, kHeight, hdc[2], 0, 0, SRCCOPY);

if (!game.grayscale\_ready || !game.paused) {

GrassNode \*p = game.h->next;

while (p) {

MRTransparentBlt(hdc[0], kGrassX[p->x], kGrassY[p->y], hdc[1],

game.grass\_focus && p->id == game.grass\_focus->id ?

kGrassHighlight[p->style] : kGrass[p->style]);

if (p->picked) {

switch (p->type) {

case MUSHROOM:

MRTransparentBlt(hdc[0], kGrassX[p->x], kGrassY[p->y], hdc[1], kMushrooms[p->score - 1]);

break;

case BOMB:

MRTransparentBlt(hdc[0], kGrassX[p->x], kGrassY[p->y], hdc[1], kBomb[p->exploded ? 1 : 0]);

break;

case NOTHING:

break;

}

}

if (!p->picked) {

\_itow\_s(p->score, buffer, 10);

MRTextOut(hdc[0], kGrassX[p->x], kGrassY[p->y], buffer);

}

p = p->next;

}

MRTransparentBlt(hdc[0], player.x, player.y, hdc[1], kPlayer[player.skin \* 4 + player.direction]);

}

if (game.paused) {

if (!game.grayscale\_ready) {

BitBlt(hdc[4], 0, 0, kWidth, kBottom, hdc[0], 0, 0, SRCCOPY);

game.grayscale\_ready = true;

}

BitBlt(hdc[0], 0, 0, kWidth, kBottom, hdc[4], 0, 0, SRCCOPY);

}

MRTextOut(hdc[0], kTextsXY[0], game.player\_name);

\_itow\_s(game.score, buffer, 10);

MRTextOut(hdc[0], kTextsXY[1], buffer);

swprintf\_s(buffer, sizeof buffer / sizeof(wchar\_t), L"%02d:%02d", game.time\_left / 60, game.time\_left % 60);

MRTextOut(hdc[0], kTextsXY[2], buffer);

if (!game.paused)

MRBitBlt(hdc[0], kButtonsXY[0], hdc[1], kButtons[5]);

if (game.button\_on\_click)

MRBitBlt(hdc[0], kButtonsXY[game.button\_focus], hdc[1], kButtons[game.button\_focus]);

if (game.paused && game.button\_on\_click && game.button\_focus == 0)

MRBitBlt(hdc[0], kButtonsXY[0], hdc[1], kButtons[4]);

EndBatchDraw();

}

void GetAndDispatchCommand(Game &game, Player &player) {

if (GetAsyncKeyState(VK\_LEFT) & 0x8000 || GetAsyncKeyState(65) & 0x8000)

player.SetDirection(LEFT);

if (GetAsyncKeyState(VK\_UP) & 0x8000 || GetAsyncKeyState(87) & 0x8000)

player.SetDirection(UP);

if (GetAsyncKeyState(VK\_RIGHT) & 0x8000 || GetAsyncKeyState(68) & 0x8000)

player.SetDirection(RIGHT);

if (GetAsyncKeyState(VK\_DOWN) & 0x8000 || GetAsyncKeyState(83) & 0x8000)

player.SetDirection(DOWN);

if (GetAsyncKeyState(VK\_SPACE) & 0x8000)

game.PickMushroom();

if (GetAsyncKeyState(VK\_RETURN) & 0x8000) {

game.HandleReturnKey();

}

MOUSEMSG message;

if (MouseHit()) {

message = GetMouseMsg();

switch (message.uMsg) {

case WM\_MOUSEMOVE:

game.button\_focus = GetGameButtonFocus(message.x, message.y);

if (game.button\_focus == -1)

game.button\_on\_click = false;

MRSetCursor(game.button\_focus);

break;

case WM\_LBUTTONDOWN:

game.button\_on\_click = game.button\_focus != -1 ? true : false;

break;

case WM\_LBUTTONUP:

game.button\_on\_click = false;

switch (game.button\_focus) {

case 0:

game.paused = !game.paused;

game.grayscale\_ready = false;

break;

case 1:

SaveGameToFile(game, player);

break;

case 2:

game.ClearGrass();

game.grayscale\_ready = false;

break;

case 3:

game.ExitGame(false);

break;

default:

break;

}

break;

}

}

}

void GetGrassFocus(Game &game, Player &player) {

const int kX[4] = { 100,300,500,700 };

const int kY[3] = { 81,243,405 };

GrassNode \*p = game.h->next;

while (p) {

if (pow(player.x - kX[p->x], 2) + pow(player.y - kY[p->y], 2) < 70 \* 70) {

game.grass\_focus = p;

return;

}

p = p->next;

}

game.grass\_focus = nullptr;

}

int GetGameButtonFocus(int x, int y) {

const int kButtons[4][2] = { { 55,542 },{ 135,542 },{ 215,542 },{ 295,542 } };

for (int i = 0; i < 4; i++) {

if (pow(x - kButtons[i][0], 2) + pow(y - kButtons[i][1], 2) < 32 \* 32)

return i;

}

return -1;

}

GrassNode::GrassNode(int i) : id(i) {

srand(clock());

picked = false;

exploded = false;

time\_picked = 0;

next = nullptr;

if (id == -1) {

type = NOTHING;

return;

}

type = rand() % 3;

style = rand() % 3;

score = rand() % 9 + 1;

do {

x = rand() % 4;

y = rand() % 3;

} while (grid[y][x] == 1);

grid[y][x] = 1;

Sleep(1);

}

int GrassNode::grid[3][4] = { 0 };

Game::Game() {

time\_left = 0;

init\_num = 5;

num\_at\_a\_time = 1;

interval = 10;

grass\_num = 0;

last\_id = 0;

h = nullptr;

grass\_focus = nullptr;

Reset();

}

void Game::InitGrass() {

GrassNode \*p, \*s;

p = h = new GrassNode(-1);

for (int i = 0; i < init\_num; i++) {

s = new GrassNode(i);

p->next = s;

p = s;

}

grass\_num = init\_num;

last\_id = init\_num;

}

void Game::NewGrass() {

GrassNode \*p = h;

while (p->next) {

p = p->next;

}

for (int i = 0; i < num\_at\_a\_time && grass\_num < 12; i++) {

p->next = new GrassNode(++last\_id);

grass\_num++;

p = p->next;

}

}

void Game::DeleteGrassById(int id) {

GrassNode \*p = h, \*s;

while (p) {

if (p->next->id == id) {

s = p->next;

p->next = p->next->next;

grass\_num--;

GrassNode::grid[s->y][s->x] = 0;

delete s;

return;

}

p = p->next;

}

}

void Game::ClearGrass() {

GrassNode \*p = h, \*q = h->next;

while (q) {

delete p;

p = q;

q = q->next;

}

delete p;

h = new GrassNode(-1);

grass\_num = 0;

last\_id = 0;

ZeroMemory(GrassNode::grid, sizeof GrassNode::grid);

}

void Game::PickMushroom() {

if (!grass\_focus)

return;

if (grass\_focus->picked)

return;

grass\_focus->picked = true;

grass\_focus->time\_picked = clock();

if (grass\_focus->type == MUSHROOM)

score += grass\_focus->score;

}

void Game::GameTimer() {

static clock\_t old = clock();

if (clock() - old >= 1000 \* 2)

old = clock();

if (clock() - old >= 1000) {

old = clock();

time\_left--;

}

if (time\_left == -1)

ExitGame(true);

}

void Game::GrassTimer() {

static clock\_t old = clock();

if (clock() - old >= interval \* 1000 \* 2) {

old = clock();

}

if (clock() - old >= interval \* 1000) {

old = clock();

NewGrass();

}

GrassNode \*p = h;

while (p) {

if (p->next && p->next->picked) {

if (clock() - p->next->time\_picked >= 2 \* 1000) {

DeleteGrassById(p->next->id);

}

if (grass\_focus && grass\_focus->id == p->next->id && p->next->type == BOMB && !p->next->exploded && clock() - p->next->time\_picked >= 5 \* 100) {

p->next->time\_picked = clock();

p->next->exploded = true;

score -= grass\_focus->score;

}

}

p = p->next;

}

}

void Game::SaveScoreToLeaderboard() {

wchar\_t path[260];

GetModuleFileName(nullptr, path, sizeof path / sizeof(wchar\_t));

PathRemoveFileSpec(path);

wcscat\_s(path, L"\\leaderboard.txt");

wfstream f;

f.open(path, ios\_base::app);

if (!f.is\_open())

return;

f.imbue(locale(locale::empty(), new codecvt\_utf8<wchar\_t>));

time\_t now = time(nullptr);

tm tstruct;

localtime\_s(&tstruct, &now);

wchar\_t buffer[11];

wcsftime(buffer, sizeof buffer / sizeof(wchar\_t), L"%Y/%m/%d", &tstruct);

f << player\_name << '\t' << score << '\t' << buffer << endl;

f.close();

}

void Game::ExitGame(bool timeout) {

switch (ShowExitGameDialog(score, timeout)) {

case IDYES:

SaveScoreToLeaderboard();

case IDNO:

on\_exit = 1;

ClearGrass();

break;

case IDCANCEL:

break;

}

}

void Game::HandleReturnKey() {

static clock\_t old = clock();

if (clock() - old >= 200) {

old = clock();

paused = !paused;

grayscale\_ready = false;

}

}

void Game::Reset() {

score = 0;

button\_focus = -1;

button\_on\_click = false;

paused = true;

grayscale\_ready = false;

on\_exit = false;

}

int CALLBACK InputDialog(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) {

switch (message) {

case WM\_INITDIALOG:

OnInitInputDialog(hDlg);

return 1;

case WM\_COMMAND:

switch (LOWORD(wParam)) {

case IDOK:

GetDlgItemText(hDlg, IDC\_EDIT1, temp\_name, 11);

for (int i = 0; i < 4; i++)

temp\_num[i] = GetDlgItemInt(hDlg, IDC\_EDIT2 + i, nullptr, 0);

temp\_num[4] = SendMessage(GetDlgItem(hDlg, IDC\_TRACKBAR1), TBM\_GETPOS, 0, 0);

temp\_num[5] = SendMessage(GetDlgItem(hDlg, IDC\_COMBOBOXEX1), CB\_GETCURSEL, 0, 0);

EndDialog(hDlg, LOWORD(wParam));

return 1;

case IDCANCEL:

EndDialog(hDlg, LOWORD(wParam));

return 1;

}

return 0;

case WM\_CLOSE:

ImageList\_Destroy((HIMAGELIST)GetWindowLongPtr(hDlg, GWLP\_USERDATA));

DestroyWindow(hDlg);

}

return 0;

}

int CALLBACK LeaderboardDialog(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam) {

static wchar\_t data[50][3][11];

switch (message) {

case WM\_INITDIALOG:

OnInitLeaderboardDialog(hDlg, data);

return 1;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || IDCANCEL) {

EndDialog(hDlg, LOWORD(wParam));

return 1;

}

case WM\_NOTIFY:

switch (((LPNMHDR)lParam)->code) {

case LVN\_GETDISPINFO:

NMLVDISPINFO\* p;

p = (NMLVDISPINFO\*)lParam;

p->item.pszText = data[p->item.iItem][p->item.iSubItem - 1];

break;

case LVN\_COLUMNCLICK:

static bool descending = false;

static int n = ListView\_GetItemCount(GetDlgItem(hDlg, IDC\_LISTVIEW1));

LPNMLISTVIEW pl = (LPNMLISTVIEW)lParam;

SortData(data, n, pl->iSubItem - 1, !descending);

descending = !descending;

ListView\_RedrawItems(GetDlgItem(hDlg, IDC\_LISTVIEW1), 0, n);

break;

}

return 1;

}

return 0;

}

void SortData(wchar\_t data[50][3][11], int n, int item, bool descending) {

if (item != 1)

return;

wchar\_t temp[3][11];

int a, b, i, j;

for (i = n - 1; i >= 1; i--) {

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

a = \_wtoi(data[j][item]);

b = \_wtoi(data[j + 1][item]);

if (descending ? a < b : a > b) {

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

wcscpy\_s(temp[k], data[j][k]);

wcscpy\_s(data[j][k], data[j + 1][k]);

wcscpy\_s(data[j + 1][k], temp[k]);

}

}

}

}

}

bool OnInitInputDialog(HWND hDlg) {

INITCOMMONCONTROLSEX iccx;

iccx.dwSize = sizeof(INITCOMMONCONTROLSEX);

iccx.dwICC = ICC\_UPDOWN\_CLASS | ICC\_PROGRESS\_CLASS | ICC\_USEREX\_CLASSES;

if (!InitCommonControlsEx(&iccx))

return false;

const int kRanges[4][2] = { { 1,3600 },{ 1,12 },{ 1,11 },{ 1,10 } };

const int kDefaults[4] = { 60,4,1,2 };

for (int i = 0; i < 4; i++) {

HWND hUpdown = CreateWindowEx(0, UPDOWN\_CLASS, nullptr,

WS\_CHILD | WS\_VISIBLE | UDS\_ALIGNRIGHT | UDS\_ARROWKEYS | UDS\_SETBUDDYINT | UDS\_WRAP,

0, 0, 0, 0, hDlg, nullptr, GetModuleHandle(nullptr), nullptr);

SendMessage(hUpdown, UDM\_SETBUDDY, (WPARAM)GetDlgItem(hDlg, IDC\_EDIT2 + i), 0);

SendMessage(hUpdown, UDM\_SETRANGE32, kRanges[i][0], kRanges[i][1]);

SetDlgItemInt(hDlg, IDC\_EDIT2 + i, kDefaults[i], 0);

}

HWND hTrack = CreateWindowEx(0, TRACKBAR\_CLASS, nullptr,

WS\_CHILD | WS\_VISIBLE | TBS\_AUTOTICKS | TBS\_TOOLTIPS,

130, 195, 80, 30, hDlg, (HMENU)IDC\_TRACKBAR1, GetModuleHandle(nullptr), nullptr);

SendMessage(hTrack, TBM\_SETRANGE, 1, MAKELONG(1, 3));

SendMessage(hTrack, TBM\_SETPOS, 1, 1);

SendMessage(hTrack, TBM\_SETBUDDY, 1, (LPARAM)GetDlgItem(hDlg, IDC\_STATIC1));

SendMessage(hTrack, TBM\_SETBUDDY, 0, (LPARAM)GetDlgItem(hDlg, IDC\_STATIC2));

const int kSkinIcons[] = { IDI\_MUSHROOM, IDI\_MUSHROOM };

const int kSkinNum = \_countof(kSkinIcons);

wchar\_t kSkinNames[2][10] = { L"Ball",L"Umaru" };

HWND hComboEx = CreateWindowEx(0, WC\_COMBOBOXEX, nullptr,

CBS\_DROPDOWNLIST | WS\_CHILD | WS\_TABSTOP | WS\_VISIBLE,

120, 230, 100, 90, hDlg, (HMENU)IDC\_COMBOBOXEX1, GetModuleHandle(nullptr), nullptr);

HIMAGELIST hImageList = ImageList\_Create(16, 16, ILC\_MASK | ILC\_COLOR32, kSkinNum, 0);

for (int i = 0; i < kSkinNum; i++)

ImageList\_AddIcon(hImageList, LoadIcon(GetModuleHandle(nullptr), MAKEINTRESOURCE(kSkinIcons[i])));

SendMessage(hComboEx, CBEM\_SETIMAGELIST, 0, (LPARAM)hImageList);

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

COMBOBOXEXITEM item = { 0 };

item.mask = CBEIF\_TEXT | CBEIF\_IMAGE | CBEIF\_SELECTEDIMAGE;

item.iItem = i;

item.iImage = i;

item.iSelectedImage = i;

item.pszText = kSkinNames[i];

SendMessage(hComboEx, CBEM\_INSERTITEM, 0, (LPARAM)&item);

}

SendMessage(hComboEx, CB\_SETCURSEL, 0, 0);

SetWindowLongPtr(hDlg, GWLP\_USERDATA, (LONG\_PTR)hImageList);

return true;

}

bool OnInitLeaderboardDialog(HWND hDlg, wchar\_t data[50][3][11]) {

INITCOMMONCONTROLSEX iccx;

iccx.dwSize = sizeof(INITCOMMONCONTROLSEX);

iccx.dwICC = ICC\_LISTVIEW\_CLASSES;

if (!InitCommonControlsEx(&iccx))

return false;

HWND hListview = CreateWindowEx(0, WC\_LISTVIEW, nullptr,

WS\_CHILD | LVS\_REPORT | WS\_VISIBLE,

16, 16, 248, 270, hDlg, (HMENU)IDC\_LISTVIEW1, GetModuleHandle(nullptr), nullptr);

ListView\_SetExtendedListViewStyle(hListview, LVS\_EX\_FULLROWSELECT | LVS\_EX\_GRIDLINES | LVS\_EX\_HEADERDRAGDROP | LVS\_EX\_DOUBLEBUFFER);

wchar\_t kHeaders[4][3] = { L"",L"玩家",L"分数",L"日期" };

const int kColumnWidth[4] = { 0,100,50,80 };

LVCOLUMN column;

column.mask = LVCF\_FMT | LVCF\_WIDTH | LVCF\_TEXT;

column.fmt = LVCFMT\_LEFT;

for (int i = 0; i < 4; i++) {

column.iSubItem = i;

column.pszText = kHeaders[i];

column.cx = kColumnWidth[i];

ListView\_InsertColumn(hListview, i, &column);

}

wchar\_t path[260];

GetModuleFileName(nullptr, path, sizeof path / sizeof(wchar\_t));

PathRemoveFileSpec(path);

wcscat\_s(path, L"\\leaderboard.txt");

wfstream f;

f.open(path, ios\_base::in);

if (!f.is\_open())

return false;

f.imbue(locale(locale(), new codecvt\_utf8<wchar\_t>));

LVITEM item;

item.mask = LVIF\_TEXT | LVIF\_STATE;

item.stateMask = 0;

item.iSubItem = 0;

item.state = 0;

for (int i = 0; !f.eof() && i < 50; i++) {

f >> data[i][0] >> data[i][1] >> data[i][2];

item.pszText = L"";

item.iItem = i;

if (data[i][0][0])

ListView\_InsertItem(hListview, &item);

}

f.close();

return true;

}

int ShowExitGameDialog(int score, bool timeout) {

int selected = 0;

wchar\_t buffer[30];

swprintf\_s(buffer, L"分数: %d\n是否记录分数?", score);

TASKDIALOGCONFIG config = { 0 };

TASKDIALOG\_BUTTON buttons[] = { { IDYES,L"记录" },{ IDNO,L"不记录" } };

config.cbSize = sizeof config;

config.hInstance = GetModuleHandle(nullptr);

config.hwndParent = GetHWnd();

config.dwCommonButtons = timeout ? 0 : TDCBF\_CANCEL\_BUTTON;

config.dwFlags = TDF\_ALLOW\_DIALOG\_CANCELLATION | TDF\_USE\_COMMAND\_LINKS;

config.pButtons = buttons;

config.cButtons = \_countof(buttons);

config.pszWindowTitle = kMushroom;

config.pszMainIcon = TD\_WARNING\_ICON;

config.pszMainInstruction = timeout ? L"游戏结束！" : L"退出游戏";

config.pszContent = buffer;

TaskDialogIndirect(&config, &selected, nullptr, nullptr);

return selected;

}

void ShowHelpDialog() {

wchar\_t buffer[30], copyright[110];

mbstowcs\_s(nullptr, buffer, \_\_DATE\_\_ " " \_\_TIME\_\_, sizeof buffer / sizeof(wchar\_t));

swprintf\_s(copyright, sizeof copyright / sizeof(wchar\_t), L"版本 %s (%s) %s\n© 2016 刘云 15071018", kVersion, buffer, kBuildType);

TASKDIALOGCONFIG config = { 0 };

config.cbSize = sizeof config;

config.hInstance = GetModuleHandle(nullptr);

config.hwndParent = GetHWnd();

config.dwCommonButtons = TDCBF\_OK\_BUTTON;

config.dwFlags = TDF\_ALLOW\_DIALOG\_CANCELLATION;

config.pszWindowTitle = L"帮助";

config.pszMainIcon = MAKEINTRESOURCE(IDI\_MUSHROOM);

config.pszMainInstruction = L"这是一个有趣的采蘑菇游戏！";

config.pszContent = L"使用 ↑↓←→ 或 WASD 键控制移动方向，按空格键尝试采蘑菇。\n采到蘑菇得分，采到炸弹减分。分数随机，范围在 1 到 9 之间。踩到炸弹后有 0.5 秒的躲避时间。\n点击清空按钮可将游戏区所有的草、蘑菇、炸弹清除。\n点击保存按钮保存游戏状态，可在主菜单重新载入。";

config.pszFooterIcon = TD\_INFORMATION\_ICON;

config.pszFooter = copyright;

TaskDialogIndirect(&config, nullptr, nullptr, nullptr);

}

wfstream GetFileStream(int mode, wchar\_t\* filename) {

wchar\_t filepath[260] = { 0 };

OPENFILENAME ofn;

ZeroMemory(&ofn, sizeof ofn);

ofn.lStructSize = sizeof ofn;

ofn.hwndOwner = GetHWnd();

ofn.lpstrFile = filepath;

ofn.nMaxFile = sizeof filepath / sizeof(wchar\_t);

ofn.lpstrFilter = L"所有文件(\*.\*)\0\*.\*\0采蘑菇存档文件(\*.mrs)\0\*.mrs\0";

ofn.nFilterIndex = 2;

ofn.Flags = OFN\_PATHMUSTEXIST;

if (filename) {

wcscpy\_s(filepath, filename);

ofn.lpstrFileTitle = filename;

ofn.nMaxFileTitle = 30;

}

wfstream f;

switch (mode) {

case 0:

ofn.Flags |= OFN\_FILEMUSTEXIST;

ofn.lpstrTitle = L"载入游戏";

if (GetOpenFileName(&ofn) == 1) {

f.open(filepath, ios\_base::in);

}

break;

case 1:

ofn.Flags |= OFN\_OVERWRITEPROMPT;

ofn.lpstrDefExt = L"mrs";

ofn.lpstrTitle = L"保存游戏";

if (GetSaveFileName(&ofn) == 1) {

f.open(filepath, ios\_base::out);

}

break;

}

f.imbue(locale(locale::empty(), new codecvt\_utf8<wchar\_t>));

return f;

}

void CreateGrayscaleBitmap(HDC hdc) {

PBITMAPINFO p = (PBITMAPINFO)LocalAlloc(LPTR, sizeof(BITMAPINFOHEADER) + (1 << 8) \* sizeof(RGBQUAD));

p->bmiHeader.biSize = sizeof(BITMAPINFOHEADER);

p->bmiHeader.biWidth = kWidth;

p->bmiHeader.biHeight = kBottom;

p->bmiHeader.biPlanes = 1;

p->bmiHeader.biBitCount = 8;

p->bmiHeader.biCompression = BI\_RGB;

p->bmiHeader.biSizeImage = p->bmiHeader.biWidth \* p->bmiHeader.biHeight;

p->bmiHeader.biClrUsed = 1 << 8;

p->bmiHeader.biClrImportant = 0;

for (SIZE\_T i = 0; i < 1 << 8; i++) {

p->bmiColors[i].rgbRed = (BYTE)i;

p->bmiColors[i].rgbGreen = (BYTE)i;

p->bmiColors[i].rgbBlue = (BYTE)i;

}

HBITMAP bitmap = CreateDIBSection(hdc, p, 0, DIB\_RGB\_COLORS, nullptr, 0);

SelectObject(hdc, bitmap);

}

void MRSetCursor(int focus) {

static HCURSOR arrow = LoadCursor(nullptr, IDC\_ARROW);

static HCURSOR hand = LoadCursor(nullptr, IDC\_HAND);

SetClassLong(GetHWnd(), GCL\_HCURSOR, focus != -1 ? (long)hand : (long)arrow);

}