**高级语言程序设计**

**课设报告**

**题 目 接馅饼**

**学 号 15071025**

**姓 名 陈仕玺**

**指导教师 诸海生**

**提交日期**

**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](#_Toc450111024)

[1.1 功能需求 3](#_Toc450111025)

[1.1.1 箱子 3](#_Toc450111026)

[1.1.2 按钮 3](#_Toc450111027)

[1.2数据需求 3](#_Toc450111028)

[1.3界面需求 4](#_Toc450111029)

[1.4 开发与运行环境需求 4](#_Toc450111030)

[1.5 其他方面需求 5](#_Toc450111031)

[2 概要设计 5](#_Toc450111032)

[2.1 主要数据结构 5](#_Toc450111033)

[2.2 程序总体结构 5](#_Toc450111034)

[2.3 子模块设计 8](#_Toc450111035)

[3 详细设计 11](#_Toc450111036)

[4 测试 14](#_Toc450111037)

[5 用户手册 18](#_Toc450111038)

[6.1 课程设计总结 19](#_Toc450111039)

[6.2 对本课程意见与建议 19](#_Toc450111040)

[6.3附件：程序源代码 20](#_Toc450111041)

# 

# 1 需求分析

完成一个简单的接馅饼游戏程序，实现下述功能。

## 功能需求

从屏幕上方水平随机位置匀速掉下箱子，箱子可能是蘑菇（占大约2/3），也可能是毒蘑菇（占大约1/3），如果是蘑菇，每过1.3s，则变为蘑菇，但一闪后又恢复为箱子，毒蘑菇也类似变化。屏幕下方有一个马里奥，随键盘操作横向运动，接到一个蘑菇则加10分，如果蘑菇未接到后落地则减10分（可出现负值）；接到毒蘑菇，马里奥死亡，扣100分，出现新的马里奥，最多有3条命。3条命用尽，则游戏结束。

### 箱子

主箱子的横坐标随机生成，两个箱子之前的间隔时间根据游戏难度随机在一定范围内确定。箱子的下落速度相同。

### 按钮

“Play” 按钮单击后开始游戏，并变为 “Pause” 按钮，单击之可以暂停游戏并再次变为“Play”。

“Stop” 按钮单击后停止当前局的游戏，并将复位到游戏开始前的状态。

“Exit” 按钮单击后退出程序。

“Settings”按钮单击后可以设置游戏难度。

“Load Score” 按钮单击后尝试从磁盘读取当前用户名以前的游戏记录，如果有超过当前最高分的记录则显示在主面板上的最高分中。

“Save Score” 按钮只能在游戏停止时点击，单击后可以将刚才一局游戏的得分存盘到对应用户名的游戏记录中，便于以后读取。

“Load Game” 按钮单击后可以从指定文件中读取以前保存的游戏状态，若当前游戏进行中则会提示是否放弃或先保存当前游戏。

“Save Game” 按钮单击后可以将当前游戏状态保存。

* + 1. **信息显示**

窗口最上方为信息显示区，有五项内容，分别为用户名、本局剩余时间、本局得分、剩余生命数和历史最高分。

一局游戏都没有进行过时，历史最高分不可用，显示为“N/A”。

剩余时间和当前得分会在游戏中动态变化。

* + 1. **游戏设置**

单击设置按钮可以弹出游戏难度控制面板，不同级别蘑菇的出现频率不同，下落速度不同，显示为原形的时间不同。

## 1.2数据需求

输入数据：用户名、存盘文件名。

从文件输入的数据：历史比分、游戏状态数据（用户名、剩余时间、比分、剩余生命、马里奥位置、小球位置、游戏难度等）。

输出数据：用户名、本局剩余时间、本局得分、剩余生命数和历史最高分。

## 1.3界面需求

如图所示：上方为信息显示区，显示用户名、本局剩余时间、本局得分、剩余生命数和历史最高分。



## 1.4 开发与运行环境需求

Visual Studio 2015

Windows XP 以上系统

.NET Framework 4.0 以上

C++ 运行库版本14以上

屏幕分辨率800\*600以上

EasyX 2015-10-15 更新版

## 1.5 其他方面需求

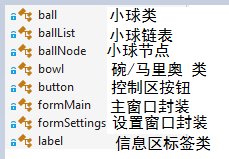
由于会产生存档，故不能在只读存储设备，如光盘，上直接运行。

# 2 概要设计

## 2.1 主要数据结构



类：

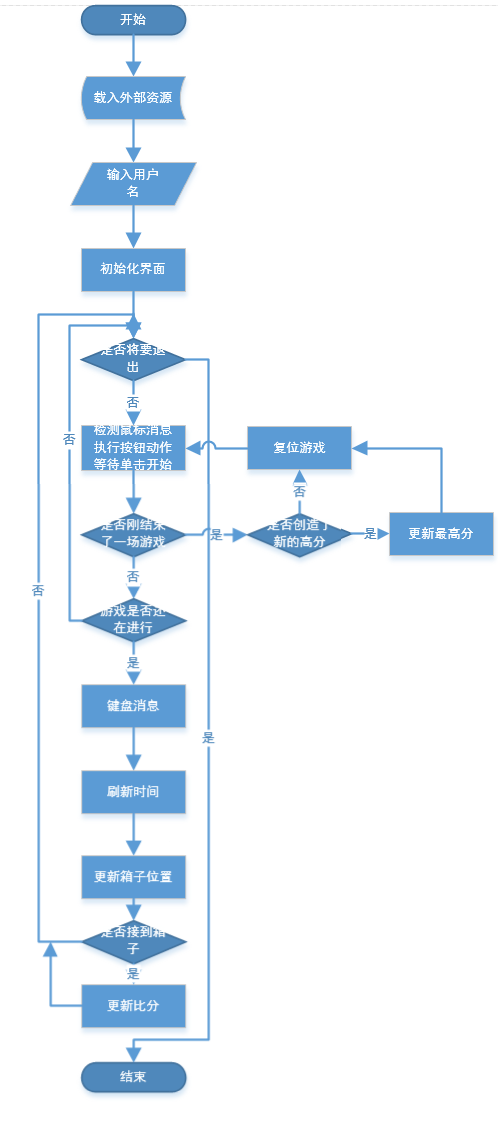


## 程序总体结构

2.2.1 模块调用图

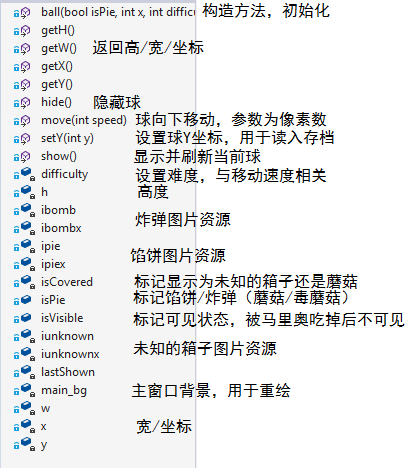


2.2.2主程序流程图

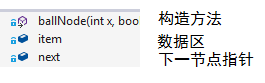


## 2.3 子模块设计

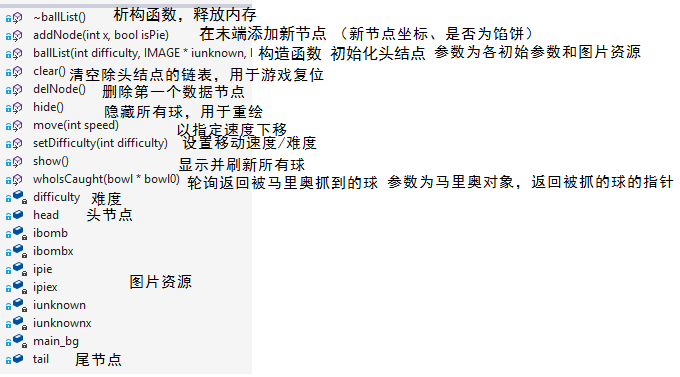
小球类：



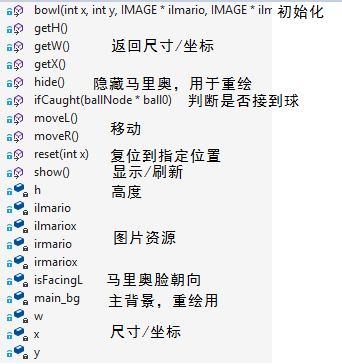
小球节点类：



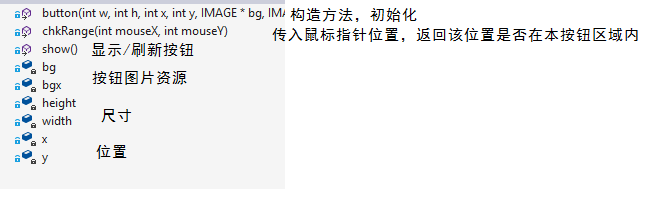
小球链表类：



马里奥/碗类：



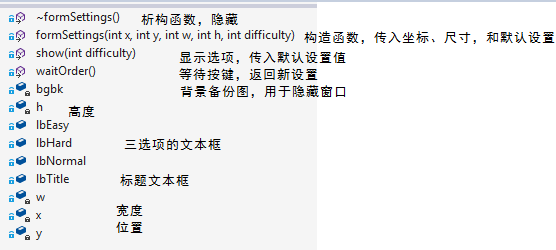
按钮类：



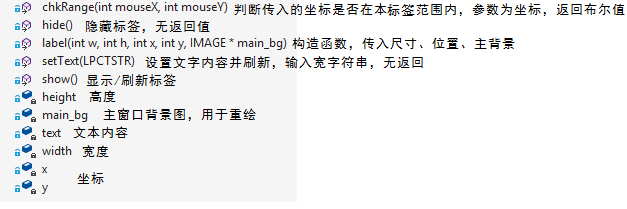
主窗口封装：



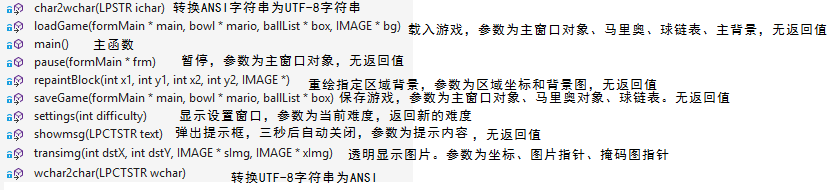
设置窗口类：



文字标签类：



全局函数：



# 3 详细设计

具体代码含义在源代码中有详尽的注释。

**3.1 鼠标消息处理**

循环检测鼠标消息：如果检测到按键，则将按键时指针位置分别传入几个按钮检测其是否在本按钮区域内。若是，则判断为按下了本按钮。



**3.2 键盘消息处理**

不应使用 getch 获取键盘消息，此函数只有在按下并弹起按键后才能触发。会造成响应延迟。键盘消息应检测按键按下状态。通过 GetAsyncKeyState(VK\_LEFT) & 0x8000 完成按键按下状态的检测。



**3.3 绘图时重叠对象的处理**

对象重叠时，如果分别处理前后层对象，会导致擦除前层对象时将绘制好的后层对象也擦掉。妥善的处理方式是整体擦除后分别重绘。

如游戏复位时：

boxes.clear();

repaintBlock(0, 0, 800, 530, &bg);

mario.reset(400);

frMain.setLife(3);

frMain.setTime(200);

frMain.setScore(0);

frMain.refreshLabels();

frMain.btPlay.show();

isPlaying = false;

gameOver = false;

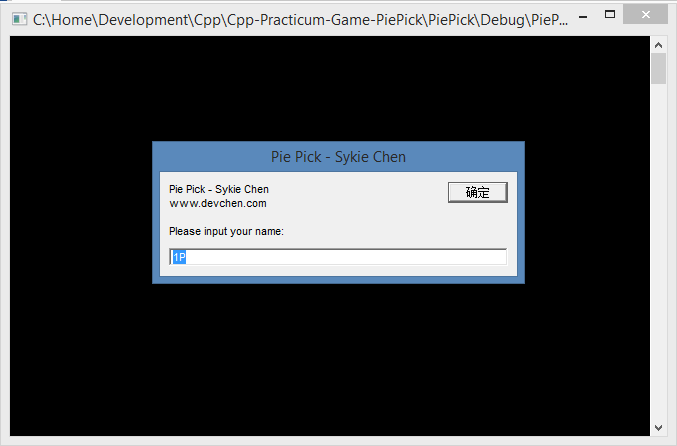
ballBreak = 0;

# 4 测试

* 1. **输入用户名**

测试目的：检测用户名输入是否正常

测试方法：输入空用户名，必须重新输入。输入带空格用户名，正常。输入超长用户名（30字符），会导致越界并退出游戏。



* 1. **开始/结束游戏**

测试目的：测试游戏控制循环。

测试方法：单击Play/Pause/Stop 按钮，游戏正常开始/暂停/重置，Play/Pause 按钮正常切换。



* 1. **难度设置**

测试目的：测试弹出设置框及相关设置参数是否正常。

测试方法：单击设置按钮，弹出设置框；单击其中选项，选项正常更新。回到游戏后难度发生相应变化。



* 1. **保存/读取最高分**

测试目的：测试文件读写及比分读入/写入。

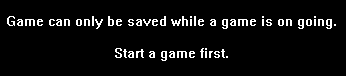
测试方法：游戏状态单击保存比分，弹窗提示必须游戏结束后再保存；非游戏状态保存比分，正常。任意状态读入比分，当前用户保存过比分时正常读入并显示在最高分区域，当前用户未保存过时弹窗提示未找到历史比分数据。

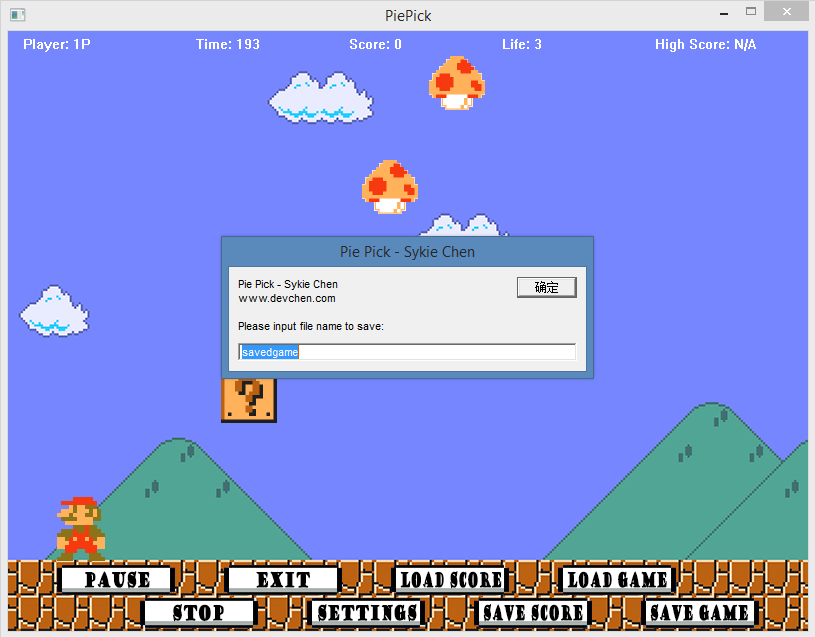


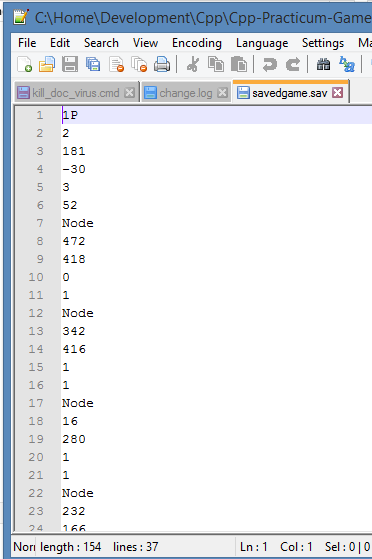
* 1. **保存游戏存档**

测试目的：测试文件读写及游戏当前状态保存。

测试方法：非游戏中单击保存，提示必须先开始游戏。游戏中保存，提示输入存档文件名，并成功写入文件。用文本编辑器打开存档文件，各项数据正常。



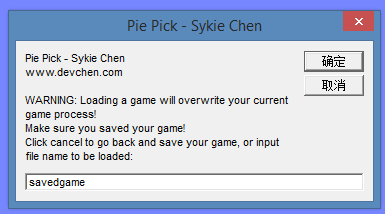


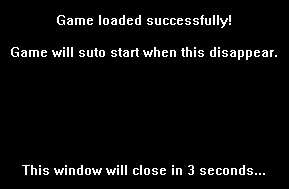


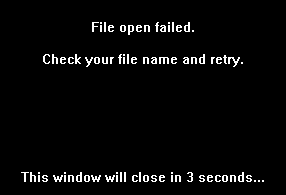
* 1. **读取游戏存档**

测试目的：测试文件读取及游戏状态控制。

测试方法：单击读取存档按钮，弹窗警告读取存档会重置当前游戏，此时单击取消回到游戏。输入存档名并确定后，若存档存在，则提示正常读入，游戏暂停三秒后继续进行。若存档不存在，提示检查存档名是否正确。







# 5 用户手册

1. 游戏介绍：

本程序是一个马里奥主题的接馅饼游戏；

进入游戏后输入用户名进入主界面。此时单击开始游戏后屏幕顶端会随机掉落箱子。箱子内可能是蘑菇或毒蘑菇，用键盘左右键控制马里奥移动。箱子间隔一定时间会瞬间变化为蘑菇或毒蘑菇，并再次变回箱子。马里奥吃到蘑菇加十分，漏掉蘑菇扣十分，吃到毒蘑菇扣一百分并损失一条生命。共有三条生命，生命值用尽或游戏超过200秒时游戏结束。

游戏结束后若当前比分高于信息区显示的最高分，最高分会更新。

1. 游戏控制：

游戏中单击暂停可以暂停游戏，此时暂停按钮变为开始按钮，单击其可以恢复游戏。

单击设置按钮可以弹出难度设置框，共有三个难度，不同难度下箱子出现的频率、显示为原型的时间、以及下落的速度均有变化。

游戏结束后单击保存分数可以将刚才一局游戏的分数写入当前用户对应的文件。

单击读取分数可以在当前用户名保存过的分数中查找最高分并更新在信息区的最高分上。若保存的分数均没有当前分数高则会提示。

游戏中单击保存游戏可以将当前游戏状态写入指定存档中。

单击读取游戏可以读取以前保存的游戏状态。读取游戏会导致当前正在进行的游戏丢失，如有需要应先保存当前游戏。

单击退出按钮退出游戏。

1. 系统环境要求：

运行于Windows XP 以上操作系统，并应安装有VC++ 2015 以上版本运行库；

运行目录必须可写，不能在只读光盘上运行；屏幕分辨率 800\*600 以上；

.NET Framework 4.0 以上。

1. 安装方法：

拷贝主程序exe与资源文件到硬盘，双击exe即可启动；

1. 注意事项：

用户名为20字符以内的任意内容，但不能包含文件系统所不容许的字符，如 \* 、 / 、 ? 等；

**6 总结提高**

## 6.1 课程设计总结

课程设计过程让我对大型程序的整体开发有了较好的把握。对于模块化设计和面向对象的封装编程有了更加深入的理解。Debug 的技术也得到了提高。

更重要的是，在本次课设十分繁杂而令人摸不到头脑，又时时变动的需求压力下，对于工作生活中甲方变更需求有了更好的容忍度和应变力。

## 6.2 对本课程意见与建议

课程设计给出的需求和老师后来的要求有一些变动，必须更改已完成的设计。

要求繁琐而不明确，让人无从下手。

课程设计前基础知识讲授不足，绝大部分知识需要自学。

## 6.3附件：程序源代码

//pie.h

#pragma once

#include <graphics.h>

#include <iostream>

#include <conio.h>

#include <time.h>

#include <fstream>

using namespace std;

void transimg(int dstX, int dstY, IMAGE \*sImg, IMAGE \*xImg);

void repaintBlock(int x1, int y1, int x2, int y2, IMAGE\*);

class button

{

private:

int width;

int height;

int x;

int y;

IMAGE\* bg;

IMAGE\* bgx;

public:

button(int w, int h, int x, int y, IMAGE\* bg, IMAGE\* bgx);

void show();

bool chkRange(int mouseX, int mouseY);

};

class label

{

private:

int x;

int y;

int width;

int height;

//int value;

wchar\_t text[30];

IMAGE\* main\_bg;

public:

label(int w, int h, int x, int y, IMAGE\* main\_bg);

void show();

void hide();

void setText(LPCTSTR);

bool chkRange(int mouseX, int mouseY);

};

class formMain

{

private:

int width;

int height;

IMAGE\* bg;

label lbUsrName, lbTime, lbScore, lbHighScore, lbLife;

int time = 200, score = 0, life = 3;

int highScore = 0;

public:

wchar\_t usrName[30];

int difficulty = 2;

button btPlay, btPause, btStop, btExit, btHs, btSave, btSet, btLg, btSg;

formMain(LPCTSTR usrName, int width, int height, IMAGE\* bg,

IMAGE\* bt\_play, IMAGE\* bt\_pause, IMAGE\* bt\_stop, IMAGE\* bt\_exit,

IMAGE\* bt\_hs, IMAGE\* bt\_save, IMAGE\* bt\_set, IMAGE\* bt\_lg, IMAGE\* bt\_sg, IMAGE\* btx);

int getTime();

void setTime(int);

int getScore();

void setScore(int);

int getHighScore();

void setHighScore(int);

int getLife();

void setLife(int);

void setUsrName(LPCTSTR);

void refreshLabels();

~formMain();

};

class ball {

private:

int x, y;

int w, h;

int difficulty;

IMAGE\* iunknown, \*iunknownx;

IMAGE\* ipie, \*ipiex;

IMAGE\* ibomb, \*ibombx;

IMAGE\* main\_bg;

bool isCovered = true;

clock\_t lastShown;

public:

bool isPie;

bool isVisible = true;

ball(bool isPie, int x, int difficulty,

IMAGE\* iunknown, IMAGE\* iunknownx,

IMAGE\* ipie, IMAGE\* ipiex,

IMAGE\* ibomb, IMAGE\* ibombx,

IMAGE\* main\_bg);

int getX();

int getY();

int getW();

int getH();

void move(int speed);

void show();

void hide();

void setY(int y);

};

class ballNode {

public:

ball item;

ballNode\* next;

ballNode(int x, bool isPie, int difficulty,

IMAGE\* iunknown, IMAGE\* iunknownx,

IMAGE\* ipie, IMAGE\* ipiex,

IMAGE\* ibomb, IMAGE\* ibombx,

IMAGE\* main\_bg);

};

class bowl {

private:

int x, y;

int w, h;

bool isFacingL = false;

IMAGE\* ilmario, \*ilmariox;

IMAGE\* irmario, \*irmariox;

IMAGE\* main\_bg;

public:

bowl(int x, int y, IMAGE\* ilmario, IMAGE\* ilmariox, IMAGE\* irmario, IMAGE\* irmariox, IMAGE\* main\_bg);

int getX();

void moveL();

void moveR();

void show();

int getW();

int getH();

void hide();

void reset(int x);

bool ifCaught(ballNode\* ball0);

};

class ballList {

private:

int difficulty;

IMAGE\* iunknown, \*iunknownx;

IMAGE\* ipie, \*ipiex;

IMAGE\* ibomb, \*ibombx;

IMAGE\* main\_bg;

public:

ballNode\* head;

ballNode\* tail;

ballList(int difficulty, IMAGE\* iunknown, IMAGE\* iunknownx,

IMAGE\* ipie, IMAGE\* ipiex,

IMAGE\* ibomb, IMAGE\* ibombx,

IMAGE\* main\_bg);

void addNode(int x, bool isPie);

void delNode();

void show();

void hide();

void move(int speed);

void clear();

void setDifficulty(int difficulty);

ballNode\* whoIsCaught(bowl\* bowl0);

~ballList();

};

class formSettings {

private:

int x, y;

int w, h;

IMAGE bgbk;

public:

label lbTitle;

label lbEasy;

label lbNormal;

label lbHard;

formSettings(int x, int y, int w, int h, int difficulty);

~formSettings();

void show(int difficulty);

int waitOrder();

};

void pause(formMain\* frm);

void showmsg(LPCTSTR text);

char\* wchar2char(LPCTSTR wchar);

wchar\_t\* char2wchar(LPSTR ichar);

int settings(int difficulty);

void saveGame(formMain\* main, bowl\* mario, ballList\* box);

void loadGame(formMain\* main, bowl\* mario, ballList\* box, IMAGE\* bg);

//ball.cpp

#include "pie.h"

ball::ball(bool isPie, int x, int difficulty,

IMAGE\* iunknown, IMAGE\* iunknownx,

IMAGE\* ipie, IMAGE\* ipiex,

IMAGE\* ibomb, IMAGE\* ibombx,

IMAGE\* main\_bg) {

this->difficulty = difficulty;

this->isPie = isPie;

this->x = x;

this->y = -90;

this->w = iunknown->getwidth();

this->h = iunknown->getheight();

this->iunknown = iunknown;

this->iunknownx = iunknownx;

this->ipie = ipie;

this->ipiex = ipiex;

this->ibomb = ibomb;

this->ibombx = ibombx;

this->main\_bg = main\_bg;

this->isCovered = true;

this->lastShown = clock();

}

int ball::getX() {

return x;

}

int ball::getY() {

return y;

}

int ball::getW() {

return w;

}

int ball::getH() {

return h;

}

void ball::move(int speed) {

y += speed;

show();

}

void ball::show() {

if (isVisible) {

//chech time cycle

if (isCovered && clock() - lastShown >= 1300) {

isCovered = false;

lastShown = clock();

}

else if (!isCovered && clock() - lastShown >= (4 - difficulty) \* 200) { //easy:600ms normal:400ms hard:200ms

isCovered = true;

lastShown = clock();

}

if (isCovered) {

transimg(x, y, iunknown, iunknownx);

}

else {

if (isPie) transimg(x, y, ipie, ipiex);

else transimg(x, y, ibomb, ibombx);

}

}

}

void ball::hide() {

repaintBlock(x, y, x + w, y + h, main\_bg);

}

void ball::setY(int y) {

this->y = y;

}

ballNode::ballNode(int x, bool isPie, int difficulty,

IMAGE\* iunknown, IMAGE\* iunknownx,

IMAGE\* ipie, IMAGE\* ipiex,

IMAGE\* ibomb, IMAGE\* ibombx,

IMAGE\* main\_bg) :

item(isPie, x, difficulty,

iunknown, iunknownx,

ipie, ipiex,

ibomb, ibombx,

main\_bg) {}

ballList::ballList(int difficulty, IMAGE\* iunknown, IMAGE\* iunknownx,

IMAGE\* ipie, IMAGE\* ipiex,

IMAGE\* ibomb, IMAGE\* ibombx,

IMAGE\* main\_bg) {

head = new ballNode(0, 0, difficulty, iunknown, iunknownx, ipie, ipiex, ibomb, ibombx, main\_bg);

head->next = NULL;

tail = head;

this->difficulty = difficulty;

this->iunknown = iunknown;

this->iunknownx = iunknownx;

this->ipie = ipie;

this->ipiex = ipiex;

this->ibomb = ibomb;

this->ibombx = ibombx;

this->main\_bg = main\_bg;

}

void ballList::addNode(int x, bool isPie) {

tail->next = new ballNode(x, isPie, difficulty, iunknown, iunknownx, ipie, ipiex, ibomb, ibombx, main\_bg);

tail = tail->next;

tail->next = NULL;

tail->item.show();

}

void ballList::delNode() {

ballNode\* t = head->next;

t->item.hide();

head->next = t->next;

delete t;

}

void ballList::show() {

ballNode\* p = head->next;

while (p != NULL) {

p->item.show();

p = p->next;

}

}

void ballList::hide() {

ballNode\* p = head->next;

while (p != NULL) {

p->item.hide();

p = p->next;

}

}

void ballList::move(int speed) {

ballNode\* p = head->next;

while (p != NULL) {

p->item.move(speed);

p = p->next;

}

}

void ballList::clear() {

ballNode\* t = head->next;

while (t != NULL) {

head->next = t->next;

delete t;

t = head->next;

}

tail = head;

}

void ballList::setDifficulty(int difficulty) {

this->difficulty = difficulty;

}

ballNode\* ballList::whoIsCaught(bowl\* bowl0) {

ballNode\* p = head->next;

while (p != NULL) {

if (p->item.getY() >= 410) {

if (bowl0->ifCaught(p)) return p;

}

p = p->next;

}

return NULL;

}

ballList::~ballList() {

ballNode\* t = head;

while (t != NULL) {

head = t->next;

delete t;

t = head;

}

}

//bowl.cpp

#include "pie.h"

bowl::bowl(int x, int y, IMAGE\* ilmario, IMAGE\* ilmariox, IMAGE\* irmario, IMAGE\* irmariox, IMAGE\* main\_bg) {

this->x = x;

this->y = y;

this->ilmario = ilmario;

this->ilmariox = ilmariox;

this->irmario = irmario;

this->irmariox = irmariox;

this->main\_bg = main\_bg;

this->w = irmario->getwidth();

this->h = irmario->getheight();

show();

}

int bowl::getX() {

return x;

}

void bowl::moveL() {

x -= 3;

isFacingL = true;

show();

}

void bowl::moveR() {

x += 3;

isFacingL = false;

show();

}

void bowl::show() {

if (isFacingL)

transimg(x, y, ilmario, ilmariox);

else

transimg(x, y, irmario, irmariox);

}

int bowl::getW() {

return w;

}

int bowl::getH() {

return h;

}

void bowl::hide() {

repaintBlock(x, y, x + w, y + h, main\_bg);

}

void bowl::reset(int x) {

this->x = x;

this->isFacingL = false;

show();

}

bool bowl::ifCaught(ballNode\* ball0) {

if (ball0->item.isVisible &&

ball0->item.getX() <= x + w &&

ball0->item.getX() + ball0->item.getW() >= x) {

ball0->item.isVisible = false;

return true;

}

return false;

}

//button.cpp

#include "pie.h"

button::button(int w, int h, int x, int y, IMAGE\* bg, IMAGE\* bgx) {

this->width = w;

this->height = h;

this->x = x;

this->y = y;

this->bg = bg;

this->bgx = bgx;

show();

}

void button::show() {

transimg(x, y, bg, bgx);

}

bool button::chkRange(int mouseX, int mouseY) {

if (mouseX >= x && mouseX <= x + width && mouseY >= y && mouseY <= y + height)

return true;

else return false;

}

//formMain.cpp

#include "pie.h"

formMain::formMain(LPCTSTR usrName, int width, int height, IMAGE\* bg,

IMAGE\* bt\_play, IMAGE\* bt\_pause, IMAGE\* bt\_stop, IMAGE\* bt\_exit,

IMAGE\* bt\_hs, IMAGE\* bt\_save, IMAGE\* bt\_set, IMAGE\* bt\_lg, IMAGE\* bt\_sg, IMAGE\* btx) :

lbUsrName(170, 20, 15, 5, bg),

lbTime(120, 20, 188, 5, bg),

lbScore(120, 20, 341, 5, bg),

lbLife(120, 20, 494, 5, bg),

lbHighScore(120, 20, 647, 5, bg),

btPlay(btx->getwidth(), btx->getheight(), 49, 534, bt\_play, btx),

btPause(btx->getwidth(), btx->getheight(), 49, 534, bt\_pause, btx),

btStop(btx->getwidth(), btx->getheight(), 132, 567, bt\_stop, btx),

btExit(btx->getwidth(), btx->getheight(), 216, 534, bt\_exit, btx),

btHs(btx->getwidth(), btx->getheight(), 383, 534, bt\_hs, btx),

btSave(btx->getwidth(), btx->getheight(), 466, 567, bt\_save, btx),

btSet(btx->getwidth(), btx->getheight(), 299, 567, bt\_set, btx),

btSg(btx->getwidth(), btx->getheight(), 633, 567, bt\_sg, btx),

btLg(btx->getwidth(), btx->getheight(), 550, 534, bt\_lg, btx)

{

initgraph(800, 600);

//paint bg

putimage(0, 0, bg);

//init boxes

wchar\_t txt[40];

wcscpy\_s(this->usrName, usrName);

swprintf\_s(txt, L"Player: %s", usrName);

lbUsrName.setText(txt);

swprintf\_s(txt, L"Time: %d", time);

lbTime.setText(txt);

swprintf\_s(txt, L"Score: %d", score);

lbScore.setText(txt);

swprintf\_s(txt, L"Life: %d", life);

lbLife.setText(txt);

swprintf\_s(txt, L"High Score: N/A");

lbHighScore.setText(txt);

//buttons

btPlay.show();

//btPause.show();

btStop.show();

btExit.show();

btHs.show();

btSave.show();

btSet.show();

btSg.show();

btLg.show();

}

formMain::~formMain() {

closegraph();

}

int formMain::getTime() {

return time;

}

void formMain::setTime(int time) {

this->time = time;

wchar\_t txt[40];

swprintf\_s(txt, L"Time: %d", time);

this->lbTime.hide();

this->lbTime.setText(txt);

}

int formMain::getScore() {

return score;

}

void formMain::setScore(int score) {

this->score = score;

wchar\_t txt[40];

swprintf\_s(txt, L"Score: %d", score);

this->lbScore.hide();

this->lbScore.setText(txt);

}

int formMain::getHighScore() {

return highScore;

}

void formMain::setHighScore(int hs) {

this->highScore = hs;

wchar\_t txt[40];

swprintf\_s(txt, L"High Score: %d", highScore);

this->lbHighScore.hide();

this->lbHighScore.setText(txt);

}

int formMain::getLife() {

return life;

}

void formMain::setLife(int life) {

this->life = life;

wchar\_t txt[40];

swprintf\_s(txt, L"Life: %d", life);

this->lbLife.hide();

this->lbLife.setText(txt);

}

void formMain::setUsrName(LPCTSTR uName) {

wcscpy\_s(usrName, uName);

wchar\_t txt[40];

swprintf\_s(txt, L"Player: %s", usrName);

this->lbUsrName.hide();

this->lbUsrName.setText(txt);

}

void formMain::refreshLabels() {

lbHighScore.show();

lbLife.show();

lbUsrName.show();

lbTime.show();

lbScore.show();

}

//formSettings.cpp

#include "pie.h"

formSettings::formSettings(int x, int y, int w, int h, int difficulty) :

lbTitle(w - 100, h / 7, x + 140, y, NULL),

lbEasy(w - 100, h / 7, x + 170, y + (h / 7), NULL),

lbNormal(w - 100, h / 7, x + 170, y + (h / 7) \* 3, NULL),

lbHard(w - 100, h / 7, x + 170, y + (h / 7) \* 5, NULL) {

this->x = x;

this->y = y;

this->w = w;

this->h = h;

getimage(&bgbk, x, y, w + 10, h + 10);

show(difficulty);

}

void formSettings::show(int difficulty) {

setfillcolor(BLACK);

solidrectangle(x, y, x + w, y + h);

lbTitle.setText(L"--Difficulty Setting--");

if (difficulty == 1) lbEasy.setText(L"[ \* ] Easy");

else lbEasy.setText(L"[ ] Easy");

if (difficulty == 2) lbNormal.setText(L"[ \* ] Normal");

else lbNormal.setText(L"[ ] Normal");

if (difficulty == 3) lbHard.setText(L"[ \* ] Hard");

else lbHard.setText(L"[ ] Hard");

}

formSettings::~formSettings() {

putimage(x, y, &bgbk);

}

int formSettings::waitOrder() {

MOUSEMSG mouseMsg = GetMouseMsg();

bool clicked = false;

int difficulty;

while (!clicked) {

if (MouseHit()) {

mouseMsg = GetMouseMsg();

if (mouseMsg.uMsg == WM\_LBUTTONDOWN) {

//clicked

if (lbEasy.chkRange(mouseMsg.x, mouseMsg.y)) {

show(1);

difficulty = 1;

clicked = true;

}

else if (lbNormal.chkRange(mouseMsg.x, mouseMsg.y)) {

show(2);

difficulty = 2;

clicked = true;

}

else if (lbHard.chkRange(mouseMsg.x, mouseMsg.y)) {

show(3);

difficulty = 3;

clicked = true;

}

}

}

Sleep(1);

}

//perform a delay to see settings clearly before it disappears

Sleep(1000);

return difficulty;

}

//functions.cpp

#include "pie.h"

void transimg(int dstX, int dstY, IMAGE \*sImg, IMAGE \*xImg) {

// Trick to display transparent images

IMAGE \*workImg = GetWorkingImage();

IMAGE \*hImg = new IMAGE; // make a copy of the origional image

Resize(hImg, sImg->getwidth(), sImg->getheight());

SetWorkingImage(hImg);

putimage(0, 0, sImg);

putimage(0, 0, xImg, SRCINVERT);

SetWorkingImage(workImg);

putimage(dstX, dstY, xImg, SRCAND);

putimage(dstX, dstY, hImg, SRCPAINT);

delete hImg;

}

void repaintBlock(int x1, int y1, int x2, int y2, IMAGE\* bg) {

putimage(x1, y1, x2 - x1, y2 - y1, bg, x1, y1);

}

void pause(formMain\* frm) {

MOUSEMSG mouseMsg = GetMouseMsg();

while (1) {

if (MouseHit()) {

mouseMsg = GetMouseMsg();

if (mouseMsg.uMsg == WM\_LBUTTONDOWN) {

//clicked

if (frm->btPause.chkRange(mouseMsg.x, mouseMsg.y)) {

frm->btPlay.show();

return;

}

}

}

Sleep(1);

}

}

void showmsg(LPCTSTR text) {

IMAGE bgbk;

getimage(&bgbk, 0, 0, 800, 600);

setfillcolor(BLACK);

solidrectangle(200, 150, 600, 450);

RECT a = { 200,250,600,400 };

drawtext(text, &a, DT\_CENTER);

a = { 200,400,600,450 };

drawtext(L"This window will close in 3 seconds...", &a, DT\_CENTER);

Sleep(3000);

putimage(0, 0, &bgbk);

}

char\* wchar2char(LPCTSTR wchar) {

char\* CStr;

size\_t len = wcslen(wchar) + 10;

size\_t converted = 0;

CStr = (char\*)malloc(len \* sizeof(char));

wcstombs\_s(&converted, CStr, len, wchar, \_TRUNCATE);

return CStr;

}

wchar\_t\* char2wchar(LPSTR ichar) {

size\_t len = strlen(ichar) + 10;

size\_t converted = 0;

wchar\_t \*WStr;

WStr = (wchar\_t\*)malloc(len \* sizeof(wchar\_t));

mbstowcs\_s(&converted, WStr, len, ichar, \_TRUNCATE);

return WStr;

}

int settings(int difficulty) {

formSettings frSett(200, 150, 400, 300, difficulty);

return frSett.waitOrder();

}

void saveGame(formMain\* fMain, bowl\* mario, ballList\* box) {

fstream oFile;

wchar\_t fileName[50];

//get file name

while (!InputBox(fileName, 50, L"Pie Pick - Sykie Chen\nwww.devchen.com\n\nPlease input file name to save:", L"Pie Pick - Sykie Chen", L"savedgame", 0, 0, true) || !wcscmp(fileName, L""));

char\* fName = wchar2char(fileName);

char\* uName = wchar2char(fMain->usrName);

strcat\_s(fName, wcslen(fileName) + 10, ".sav");

oFile.open(fName, ios::out);

if (!oFile.is\_open()) {

showmsg(L"File open failed.\n\nCheck your file name and retry.");

delete fName;

delete uName;

return;

}

oFile << uName << '\n'

<< fMain->difficulty << "\n"

<< fMain->getTime() << '\n'

<< fMain->getScore() << '\n'

<< fMain->getLife() << '\n'

<< mario->getX() << '\n';

ballNode\* p = box->head->next;

while (p != NULL) {

oFile << "Node\n"

<< p->item.getX() << '\n'

<< p->item.getY() << '\n'

<< p->item.isPie << '\n'

<< p->item.isVisible << '\n';

p = p->next;

}

wchar\_t txt[50];

swprintf\_s(txt, L"Game successfully saved to %s.sav", fileName);

showmsg(txt);

oFile.close();

delete fName;

delete uName;

}

void loadGame(formMain\* fMain, bowl\* mario, ballList\* box, IMAGE\* bg) {

fstream oFile;

wchar\_t fileName[50];

//get file name

bool ret = InputBox(fileName, 50, L"Pie Pick - Sykie Chen\nwww.devchen.com\n\nWARNING: Loading a game will overwrite your current game process!\nMake sure you saved your game!\nClick cancel to go back and save your game, or input file name to be loaded:",

L"Pie Pick - Sykie Chen", L"savedgame", 0, 0, false);

//if canceled, return

if (!ret || wcscmp(fileName, L"") == 0) return;

repaintBlock(0, 0, 800, 530, bg);

char\* fName = wchar2char(fileName);

strcat\_s(fName, wcslen(fileName) + 10, ".sav");

char uName[30];

int time, score, life, marioX, ballX, ballY, ballIsPie, ballIsVisible;

int difficulty;

oFile.open(fName, ios::in);

//check if opened

if (!oFile.is\_open()) {

showmsg(L"File open failed.\n\nCheck your file name and retry.");

delete fName;

return;

}

wchar\_t\* wUName;

oFile >> uName >> difficulty >> time >> score >> life >> marioX;

wUName = char2wchar(uName);

fMain->setUsrName(wUName);

fMain->difficulty = difficulty;

box->setDifficulty(difficulty);

fMain->setTime(time);

fMain->setLife(life);

fMain->setScore(score);

mario->reset(marioX);

box->clear();

char nodeMark[30];

oFile >> nodeMark;

wchar\_t\* wNodeMark = char2wchar(nodeMark);

while (wcscmp(wNodeMark, L"Node") == 0) {

//input balls

oFile >> ballX >> ballY >> ballIsPie >> ballIsVisible;

box->addNode(ballX, ballIsPie);

box->tail->item.isVisible = ballIsVisible;

box->tail->item.setY(ballY);

//read next

oFile >> nodeMark;

delete wNodeMark;

wNodeMark = char2wchar(nodeMark);

}

delete fName;

delete wUName;

delete wNodeMark;

showmsg(L"Game loaded successfully!\n\nGame will suto start when this disappear.");

}

//label.cpp

#include "pie.h"

label::label(int w, int h, int x, int y, IMAGE\* bg) {

this->width = w;

this->height = h;

this->x = x;

this->y = y;

this->main\_bg = bg;

//show();

}

void label::show() {

setbkmode(TRANSPARENT);

RECT r = { x,y,x + width,y + height };

drawtext(text, &r, DT\_LEFT | DT\_TOP);

}

void label::hide() {

repaintBlock(x, y, x + width, y + height, main\_bg);

}

void label::setText(LPCTSTR text) {

wcscpy\_s(this->text, text);

show();

}

bool label::chkRange(int mouseX, int mouseY) {

if (mouseX >= x && mouseX <= x + width && mouseY >= y && mouseY <= y + height)

return true;

else return false;

}

//main.cpp

#include "pie.h"

int main() {

srand((unsigned)time(NULL));

//load resources

IMAGE bg, lmario, lmariox, rmario, rmariox;

IMAGE unknown, unknownx, pie, piex, bomb, bombx;

IMAGE bt\_play, bt\_pause, bt\_save, bt\_hs, bt\_exit, bt\_stop, bt\_set, bt\_lg, bt\_sg, btx;

loadimage(&bg, L"img\\bg.jpg");

loadimage(&lmario, L"img\\lmario.bmp");

loadimage(&lmariox, L"img\\lmariox.bmp");

loadimage(&rmario, L"img\\rmario.bmp");

loadimage(&rmariox, L"img\\rmariox.bmp");

loadimage(&unknown, L"img\\unknown.bmp");

loadimage(&unknownx, L"img\\unknownx.bmp");

loadimage(&pie, L"img\\pie.bmp");

loadimage(&piex, L"img\\piex.bmp");

loadimage(&bomb, L"img\\bomb.bmp");

loadimage(&bombx, L"img\\bombx.bmp");

loadimage(&bt\_play, L"img\\bt\_play.bmp");

loadimage(&bt\_pause, L"img\\bt\_pause.bmp");

loadimage(&bt\_save, L"img\\bt\_save.bmp");

loadimage(&bt\_hs, L"img\\bt\_hs.bmp");

loadimage(&bt\_exit, L"img\\bt\_exit.bmp");

loadimage(&bt\_stop, L"img\\bt\_stop.bmp");

loadimage(&bt\_set, L"img\\bt\_set.bmp");

loadimage(&bt\_lg, L"img\\bt\_lg.bmp");

loadimage(&bt\_sg, L"img\\bt\_sg.bmp");

loadimage(&btx, L"img\\btx.bmp");

//Get username

wchar\_t usrName[20];

while (!InputBox(usrName, 20, L"Pie Pick - Sykie Chen\nwww.devchen.com\n\nPlease input your name:", L"Pie Pick - Sykie Chen", L"1P", 0, 0, true) || !wcscmp(usrName, L""));

//init

formMain frMain(usrName, 800, 600,

&bg, &bt\_play, &bt\_pause, &bt\_stop, &bt\_exit, &bt\_hs, &bt\_save, &bt\_set, &bt\_lg, &bt\_sg, &btx);

bowl mario(400, 466, &lmario, &lmariox, &rmario, &rmariox, &bg);

ballList boxes(frMain.difficulty, &unknown, &unknownx, &pie, &piex, &bomb, &bombx, &bg);

//flags

int scoreOfThisGameBackUpForSaving = 0;

bool isExiting = false;

bool isPlaying = false;

bool gameOver = false;

clock\_t kbDelay = clock();

clock\_t gameTime = 0;

clock\_t animDelay = clock();

clock\_t ballBreak = 0;

MOUSEMSG mouseMsg = GetMouseMsg();

fstream oFile;

//main loop

while (!isExiting) {

//check mouse

if (MouseHit()) {

mouseMsg = GetMouseMsg();

if (mouseMsg.uMsg == WM\_LBUTTONDOWN) {

//clicked

if (frMain.btExit.chkRange(mouseMsg.x, mouseMsg.y)) {

//clicked exit

isExiting = true;

}

else if (frMain.btPlay.chkRange(mouseMsg.x, mouseMsg.y)) {

if (!isPlaying) {

//clicked play

frMain.btPause.show();

isPlaying = true;

frMain.setLife(3);

frMain.setScore(0);

frMain.setTime(200);

}

else {

//clicked pause

frMain.btPlay.show();

pause(&frMain);

animDelay = clock();

ballBreak = 1;

}

}

else if (frMain.btStop.chkRange(mouseMsg.x, mouseMsg.y)) {

//clicked stop

if (isPlaying) gameOver = true;

}

else if (frMain.btHs.chkRange(mouseMsg.x, mouseMsg.y)) {

//high score

char\* filename = wchar2char(usrName);

//switch wchar to char

//use current username as filename

strcat\_s(filename, wcslen(usrName) + 10, ".txt");

//load score from file

wchar\_t txt[400];

oFile.open(filename, ios::in);

if (oFile.is\_open()) {

//open succeeded

int savedScore = 0;

bool isHSUpdated = false;

while (!oFile.eof()) {

oFile >> savedScore;

if (savedScore > frMain.getHighScore()) {

frMain.setHighScore(savedScore);

isHSUpdated = true;

}

}

oFile.close();

if (isHSUpdated) {

swprintf\_s(txt, L"Successfully loaded score from file %s.txt\n\nLoaded highest score %d", usrName, frMain.getHighScore());

}

else {

swprintf\_s(txt, L"Successfully loaded file %s.txt\n\nBut no score is higher than current High Score.", usrName);

}

}

else {

//no saved score

swprintf\_s(txt, L"Failed to load %s.txt\n\nYou did not save any score.", usrName);

}

showmsg(txt);

delete filename;

}

else if (frMain.btSave.chkRange(mouseMsg.x, mouseMsg.y)) {

//save game score

if (!isPlaying) {

//write to file

//switch wchar to char

char\* filename = wchar2char(usrName);

//use current username as filename

strcat\_s(filename, wcslen(usrName) + 10, ".txt");

oFile.open(filename, ios::app);

oFile << scoreOfThisGameBackUpForSaving << '\n';

oFile.close();

wchar\_t txt[400];

swprintf\_s(txt, L"Score saved to %s.txt", usrName);

showmsg(txt);

delete filename;

}

else {

showmsg(L"You can not save score while the game is on going!\n\nTry saving when game ends.");

}

}

else if (frMain.btSet.chkRange(mouseMsg.x, mouseMsg.y)) {

//settings

if (isPlaying) {

showmsg(L"You can not change settings while the game is on going!\n\nTry saving when game ends.");

}

else {

frMain.difficulty = settings(frMain.difficulty);

boxes.setDifficulty(frMain.difficulty); //set boxes' time

}

}

else if (frMain.btLg.chkRange(mouseMsg.x, mouseMsg.y)) {

//load game

loadGame(&frMain, &mario, &boxes, &bg);

frMain.refreshLabels();

wcscpy\_s(usrName, frMain.usrName);

isPlaying = true;

frMain.btPause.show();

animDelay = clock();

ballBreak = 1;

}

else if (frMain.btSg.chkRange(mouseMsg.x, mouseMsg.y)) {

//save game

if (!isPlaying) {

showmsg(L"Game can only be saved while a game is on going.\n\nStart a game first.");

}

else {

saveGame(&frMain, &mario, &boxes);

}

}

}

}

//game over?

if (gameOver) {

scoreOfThisGameBackUpForSaving = frMain.getScore();

//update highscore

if (frMain.getHighScore() < frMain.getScore()) {

frMain.setHighScore(frMain.getScore());

}

//reset game

boxes.clear();

repaintBlock(0, 0, 800, 530, &bg);

mario.reset(400);

frMain.setLife(3);

frMain.setTime(200);

frMain.setScore(0);

frMain.refreshLabels();

frMain.btPlay.show();

isPlaying = false;

gameOver = false;

ballBreak = 0;

}

else {

if (isPlaying) {

//in game

//check keyboard

if (\_kbhit()) {

//slow down the loop

if (clock() - kbDelay > 6) {

kbDelay = clock();

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

//move left

BeginBatchDraw();

mario.hide();

boxes.show();

frMain.refreshLabels();

mario.moveL();

EndBatchDraw();

}

else if (GetAsyncKeyState(VK\_RIGHT) & 0x8000) {

//move right

BeginBatchDraw();

mario.hide();

boxes.show();

frMain.refreshLabels();

mario.moveR();

EndBatchDraw();

}

}

}

//refresh time

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

gameTime = clock();

//frMain.refreshLabels();

frMain.setTime(frMain.getTime() - 1);

if (frMain.getTime() == 0) {

gameOver = true;

showmsg(L"Game Over\n\nTime up.");

}

}

//auto moves

if (clock() - animDelay >= 30) {

BeginBatchDraw();

animDelay = clock();

//release new boxes

if ((clock() - ballBreak) >= (500 + (4 - frMain.difficulty) \* 1000) || ballBreak == 0) {

ballBreak = clock() - (rand() % 1000);

//easy:2.5-3.5s normal:1.5-2.5s hard:0.5-1.5s

boxes.addNode(rand() % (800 - boxes.head->item.getW()), rand() % 3);

//random x pos, 66% possibility to be a pie

}

//hide boxes

boxes.hide();

//move boxes

boxes.move(frMain.difficulty);//the speed of ball falls

//hit mario's head

ballNode\* p = boxes.whoIsCaught(&mario);

if (p != NULL) {

//someone hit his head

if (p->item.isPie) {

//a pie hit

frMain.setScore(frMain.getScore() + 10);

}

else {

//a bomb hit

frMain.setScore(frMain.getScore() - 100);

frMain.setLife(frMain.getLife() - 1);

if (frMain.getLife() == 0) {

//dead

EndBatchDraw();

gameOver = true;

showmsg(L"Game Over\n\nLife zero.");

}

}

}

//hit bottom

if (boxes.head->next->item.getY() >= 474) {

if (boxes.head->next->item.isVisible &&

boxes.head->next->item.isPie) {

//drpooed pie -10

frMain.setScore(frMain.getScore() - 10);

}

boxes.delNode();

}

//repaint things

frMain.refreshLabels();

mario.show();

EndBatchDraw();

}

}

}

}

return 0;

}