南京信息工程大学

《面向对象程序设计-汉诺塔》
课程报告

专业：计算机科学与技术

年级：2022

姓名：许辰涛

学号：不告诉你

二0二二 年 12 月 28 日

目录

一、项目概述

1.1 主要内容

法国数学家爱德华·卢卡斯曾编写过一个印度的古老传说：在世界中心贝拿勒斯（在印度北部）的圣庙里，一块黄铜板上插着三根宝石针。印度教的主神梵天在创造世界的时候，在其中一根针上从下到上地穿好了由大到小的 64 片金片，这就是所谓的汉诺塔。不论白天黑夜，总有一个僧侣在按照下面的法则移动这些金片：一次只移动一片，不管在哪根针上，小片必须在大片上面。僧侣们预言，当所有的金片都从梵天穿好的那根针上移到另外一根针上时，世界就将在一声霹雳中消灭，而梵塔、庙宇和众生也都将同归于尽。

本项目实现了汉诺塔游戏，图形绘制使用 EsayX 实现，运行时同时显示游戏界面和控制台，控制台可输出一些操作信息，便于调试。各主要功能模块都被封装为类，体现了面向对象的程序设计思想。

本游戏用鼠标操作，用户使用鼠标左键单击按键即可。除了常驻的“主菜单”按键，ESC 键也可返回主菜单。

本游戏实现的功能如下：

（1）游戏架构设计，将行为放置于对应位置即可实现想要的效果。

（2）按键设计，包括基本按键和移动汉诺塔的按键。

（3）可显示汉诺塔，并可通过按键移动汉诺塔。

（4）有 6 个难度等级。可新建游戏，也可读取存档。有排行榜。

（5）设计了时钟，可记录玩家花费时间，也可作为游戏失败的判断标准。

（6）有游戏教学，可演示正确的移动步骤。

（7）有背景音乐，可手动控制其开关。

（8）能保存或显示玩家的游戏记录。

图 1.1 汉诺塔游戏主界面及控制台

1.2 游戏玩法

鼠标点击按键进行操作。

点击主菜单上的“游戏说明”按键可查看游戏规则。

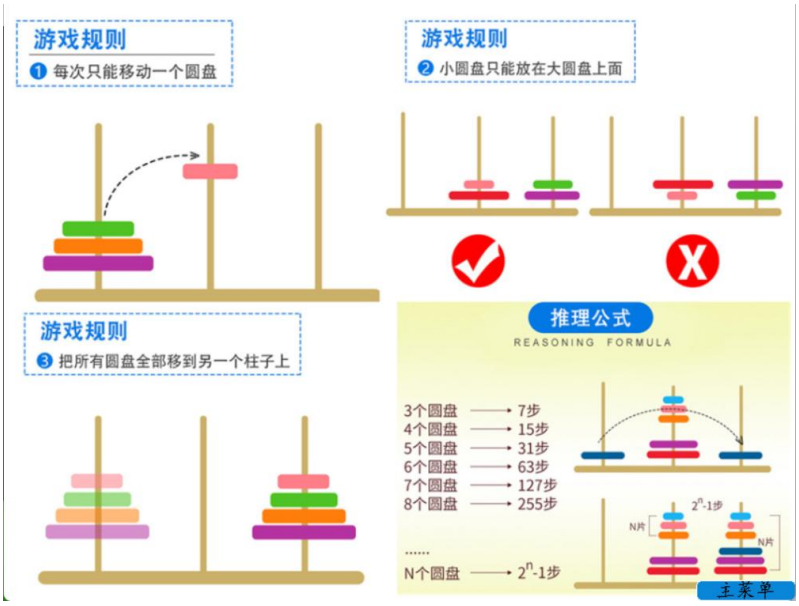

图 1.2 游戏说明界面

移动汉诺塔时，点击塔下方的棕色按键即可。图 1.2 情况表示第一个按键刚才被点击过，此时鼠标在第二个按键上。此时若点击第二个按键，则第一个塔上方的 1 层会被移动到第二个塔上。

左上角显示了游戏的剩余时间，时间用尽则游戏失败。

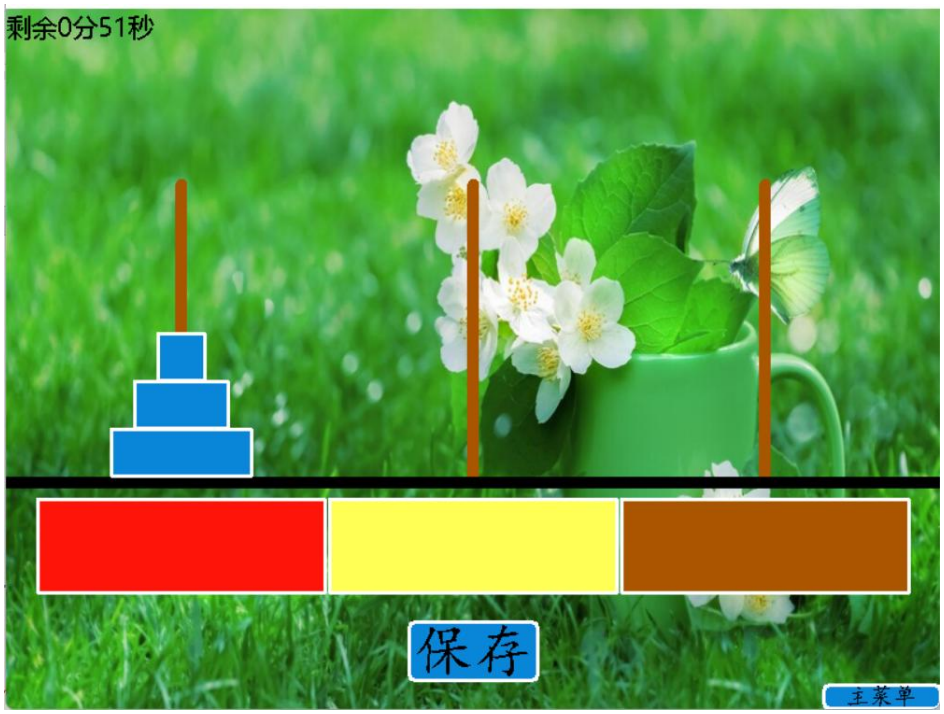

图 1.3 汉诺塔游戏界面

1.3 开发环境及配置

开发环境为 Visual Studio 2022，安装了 EsayX 图形库。使用该图形库应包含头文件 graphics.h。

为了保证能在使用 EsayX 库时正常使用中文字符，进入项目--<项目名>属性--配置属性--高级--字符集，将字符集修改为使用多字节字符集。

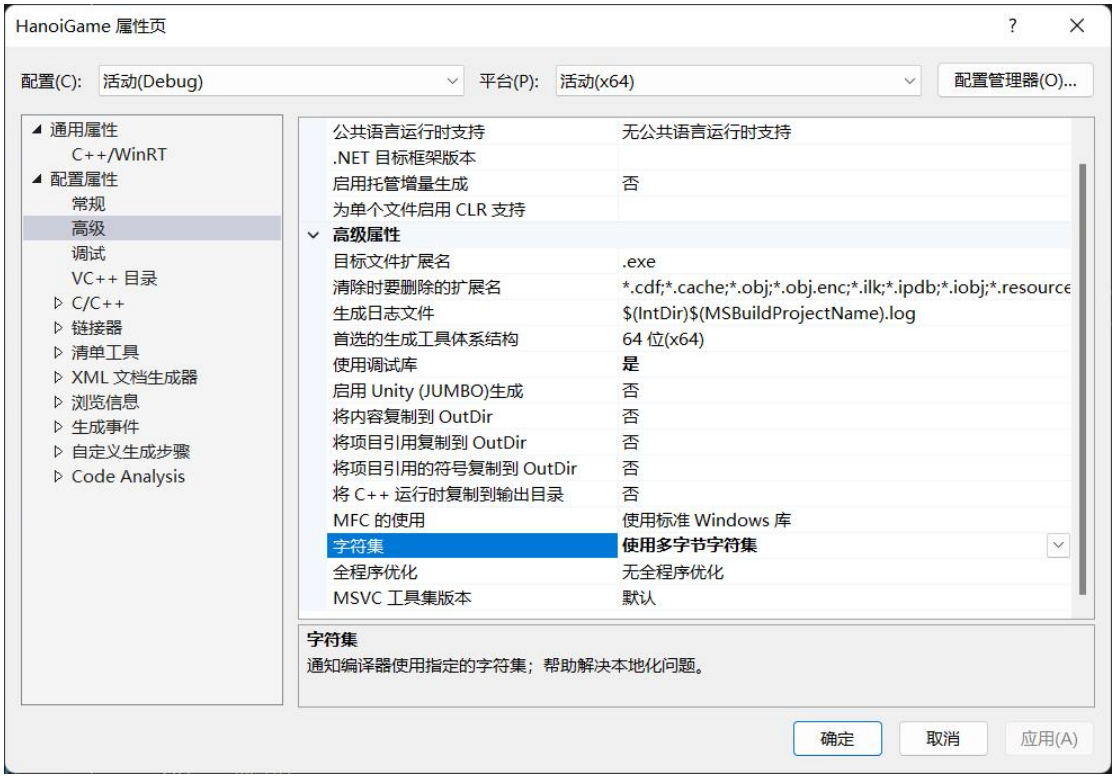

图 1.4 字符集配置

二、游戏架构设计

本项目将游戏封装为一个 Game 类，提供了两个外部接口，一个是游戏初始化 init()，另一个是游戏主循环 loop()。游戏初始化完成的是游戏启动前的准备工作（构造函数中也有），游戏主循环接收用户输入信息（鼠标,键盘），执行对应操作，同时输出游戏画面。主程序如图 2.1 所示。

```
#include "Game.h"

int main()
{
    Game G;//游戏对象
    G.init();//游戏初始化
    G.loop();//游戏循环体
    return 0;
}
```

图 2.1 main 函数中的游戏

Game 类对象中的变量在构造函数中被初始化，游戏开始前的准备工作在 init()中。游戏的运行部分在 loop()中。

```
void Game::loop()
{
    BeginBatchDraw();          //开始批量绘图
    while (op != exitGame)//游戏未结束
    {
        GameDraw();                //绘制游戏画面--这里画的图在后台
        FlushBatchDraw();          //刷新批量绘图--图在这里被刷新
        undateWithoutInput();      //与用户输入无关的信息
        undateWithInput();         //与用户输入有关的信息
    }
    EndBatchDraw();//结束批量绘图
}
```

图 2.2 Game 类的外部接口 loop()

BeginBatchDraw(),FlushBatchDraw()和 EndBatchDraw()是 EsayX 图形库中的绘图函数，使用 BeginBatchDraw()后，GameDraw()绘制的图像都会存于后台，调用 FlushBatchDraw()才会将后台的图像刷新到窗口中，EndBatchDraw()表示结束批量绘图。

Game 类中有一枚举变量 op，表示系统当前的状态，当 op 为 exitGame 时，游戏便退出，否则根据 op 的值不断地刷新图像，接收用户输入信息。

```
enum Operator {
    menu,              //主菜单
    newGame,           //新游戏难度选择
    newGame_Game,      //新游戏游戏界面
    readFile,          //读取游戏存档
    rank,              //排行榜
    tutorial,          //游戏教程难度选择
    tutorial_Game,     //游戏教程游戏界面
    GameRecord,        //查看游戏记录
    HowToPlay,         //查看游戏说明
    exitGame           //退出游戏
};
Operator op = menu;//菜单状态
```

图 2.3 枚举变量 op

undateWithoutInput()会更新与用户输入无关的数据，如游戏时间，游戏结束或游戏教程结束的判断等。

undateWithInput()会根据用户输入信息（鼠标，键盘）执行对应的操作，主要是按键事件响应。注意该函数不会像 scanf()函数一样产生阻塞效果，保证了画面的正常刷新。

GameDraw()，undateWithoutInput()和 undateWithInput()都会根据系统状态不同执行不同的操作。

```
// 与用户输入有关的信息
void Game::undateWithInput()
{
    // 获取一条鼠标或按键消息
    ExMessage msg;
    //若消息存在，则执行对应行为，否则什么也不干
    if (peekmessage(&msg, EX_MOUSE | EX_KEY)) { ... }
}
```

图 2.4 更新与用户输入有关的数据

三、主要类的设计

3.1 类概述及类间关系

各类间关系如图 3.1 所示。

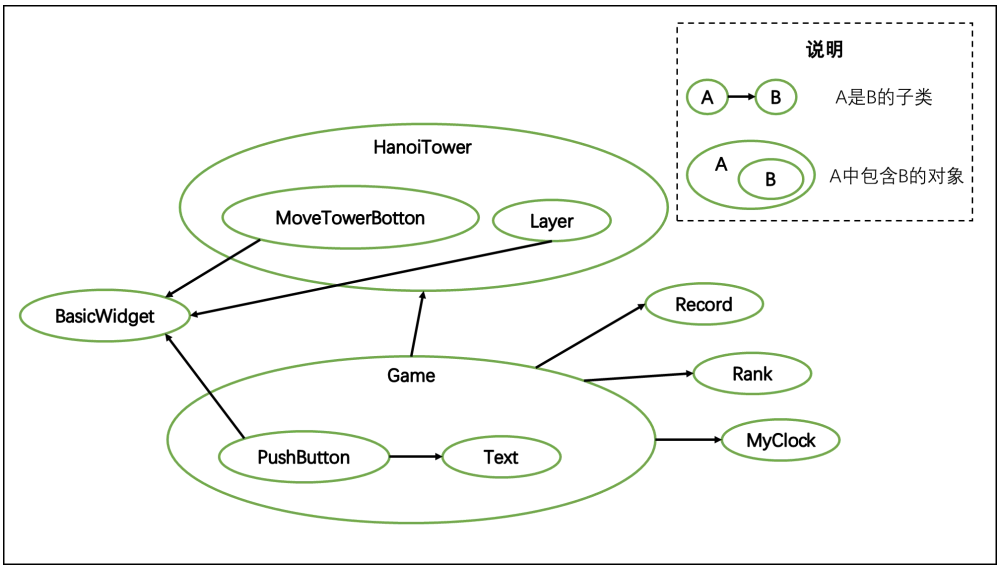

图 3.1 类之间的关系

Game 类继承了 Record 类、HanoiTower 类、Rank 类和 MyClock 类，将这四个类中的各种行为添加进游戏架构中的不同位置，再加入一些控件，就形成了游戏。HanoiTower 类包含汉诺塔相关控件，汉诺塔的显示，移动等行为都在此类中。Rank 类包含排行榜相关行为，有读写本地文件、更新排行榜等。MyClock 类包含时钟相关行为，可当作一个计时器来使用。Record 类包含游戏记录的相关行为。

BasicWidget 类是一个抽象类，是控件基类。MoveTowerButton 类、PushButton 类和 Layer 类都继承了该类，MoveTowerButton 类、PushButton 类都是按键控件，可与鼠标操作交互。Layer 类是汉诺塔的一层，它无法与鼠标操作交互，只有显示功能。

PushButton 类继承了两个类，分别为 BasicWidget 类和 Text 类，BasicWidget 类用来实现它的显示和交互行为，Text 类用来显示按键上的文字。

游戏中的动态分配存储空间都在析构函数中释放了。

3.2 BasicWidget 类

BasicWidget 类是控件基类，本项目中 MoveTowerButton 类、PushButton 类和 Layer 类都继承了该类。它的数据成员包括控件的位置和大小。可通过外部接口重新设置数据成员值，也可访问数据成员值。为便于继承，成员函数被设为保护成员（protected）。

显示控件的函数 show()是纯虚函数，在派生类中会覆写该函数。

```
class BasicWidget
{
protected:
    int m_x;//坐标x
    int m_y;//坐标y
    int m_w;//宽度
    int m_h;//高度
public:
    BasicWidget(int x, int y, int w, int h);
    int x();         //返回坐标x
    int y();         //返回坐标y
    int width();     //返回宽度
    int height();    //返回高度
    void setFixedSize(int w, int h);//重设高宽
    void move(int x, int y);        //移动位置
    virtual void show() = 0;        //纯虚函数，显示控件
};
```

图 3.2 BasicWidget 类

3.3 Layer 类

Layer 类表示汉诺塔中的一层，它简单地公有继承了 BasicWidget 类。注意该类的位置表示矩形的中心位置，而不是矩形左上角的位置，这样做可方便地使汉诺塔层居中显示在杆子上。

```
class Layer :
    public BasicWidget
{
public:
    Layer(int x, int y, int w, int h);
    void show() override;//显示层
};
```

图 3.3 Layer 类

图 3.4 Layer 显示（三层）

Layer 是一个带边框的矩形，使用 EsayX 的函数 fillrectangle()显示填充矩形。显示时先设置绘图参数，再绘图。之后不再赘述类似的绘图过程。

```
void Layer::show()
{
    //绘制层
    setfillcolor(RGB(10, 134, 217));
    setlinestyle(PS_SOLID, 3);//线条样式
    setlinecolor(WHITE);//线颜色
    fillrectangle(m_x - m_w / 2, m_y - m_h / 2, m_x + m_w / 2, m_y + m_h / 2);
}
```

图 3.5 显示 Layer

3.4 MoveTowerButton 类

MoveTowerButton 类是用来移动汉诺塔的按键。它是从 BasicWidget 类继承来的，它与 Layer 类相比，增加了按键交互功能，可通过鼠标点击按键进行操作。

eventLoop()可传入鼠标消息和键盘消息，isin()判断传入的鼠标坐标是否在矩形内，返回 true 或 false，isClicked()在 isin()基础上增加了鼠标左键是否被按下的判断。显示按键

时会根据鼠标状态进行显示。按键点击事件在 Game 类 loop()的 undateWithInput()中。

```
class MoveTowerBotton :
    public BasicWidget
{
    ExMessage m_msg;
    bool isStart = false;//该按键是否是起点
public:
    MoveTowerBotton(int x, int y, int w, int h);
    void show()override;     //显示按键
    bool isin();             //鼠标是否在当前按钮上
    bool isClicked();        //鼠标是否点击了按钮
    void eventLoop(const ExMessage& msg);//传入鼠标事件
    void setIsStart();       //该按键是起点
    void resetIsStart();     //该按键不是起点
    void traIsStart();       //是/不是起点，状态转换
};
```

图 3.6 MoveTowerButton 类

移动汉诺塔时，用户点击汉诺塔下方的按键。点击两个按键，即可实现汉诺塔的移动操作。按键原始状态为棕色，鼠标在按键上时按键会变为黄色，按键被点击时会变红色，表示被选为起点，重复点击按键可转换按键状态。该类有一数据成员 isStart，表示该按键是否是起点。

图 3.7 MoveTowerButton 类按键的不同显示状态

3.5 Text 类

游戏需要显示文字，文字用 Text 类实现。Text 类数据成员有文字位置、文字内容和文字大小，有了这些参数就可用 EsayX 将文字绘制出来。showText()调用 EsayX 的 API 将文字显示出来。显示文字与显示图形类似，先设置文字参数，再输出即可。

```
class Text
{
protected:
    int textSize = 50;  //文字大小
    std::string text;   //文字
    int text_x;         //文字坐标x
    int text_y;         //文字坐标y
public:
    Text(const std::string& arr, int textSize, int text_x = 0, int text_y = 0);
    void showText();//显示文字
};
```

图 3.8 Text 类

3.6 PushButton 类

游戏中的大部分按键都属于 PushButton 类，如图 3.9 所示。

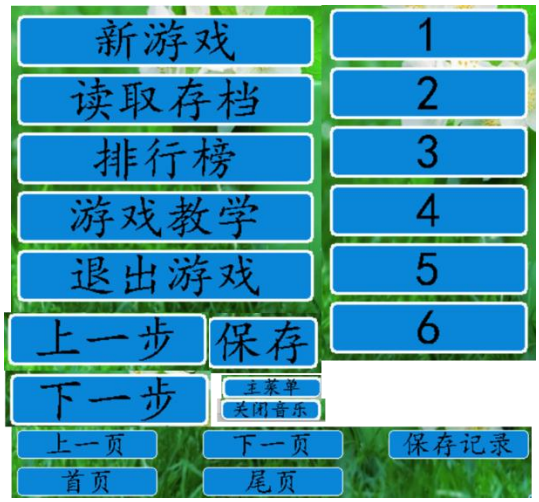

图 3.9 PushButton 类的按键

PushButton 类继承自 BasicWidget 类和 Text 类。支持鼠标操作。它增加了一个 setText() 函数，可更新按键上的文字，在打开音乐和关闭音乐时会用到。

```
class PushBotton :public BasicWidget, public Text
{
    ExMessage m_msg;//消息
public:
    PushBotton(const std::string& text = "BUTTON", int x = 0
    void setText(const std::string& text);//重新设置文字
    void show() override;    //显示按键
    bool isin();             //鼠标是否在当前按钮上
    bool isClicked();        //鼠标是否点击了按钮
    void eventLoop(const ExMessage& msg);//传入鼠标键盘消息
};
```

图 3.10 PushButton 类

3.7 HanoiTower 类

HanoiTower 类包含汉诺塔和它的行为。如图 3.11。

```
protected:
    //三个底座中心坐标
    int bottomy = 400;
    int ax = 150;
    int bx = 400;
    int cx = 650;
    //单位塔
    int towerSize = 40;
    //被选中的塔,用于操作
    int startTower = 0;
    int endTower = 0;
    //三个塔堆
    std::list<Layer*> A;
    std::list<Layer*> B;
    std::list<Layer*> C;
    //游戏教程中,解步骤指针
    int nowStep = 0;
    int i = 0;//计算解的计数器
    //下一步解
    int step[64][2] = {};
    //存档文件名
    std::string fileName;
    //移动汉诺塔的三个按键
    MoveTowerButton moveA;
    MoveTowerButton moveB;
    MoveTowerButton moveC;
    int stepNum = 0;//用户移动了几步
    int stepShift = 0;//用户移动了几步，位移
public:
    HanoiTower();
    ~HanoiTower();
    void initTower(int num);//初始化塔
    void moveTower(int start, int end);//移动塔，通过标号
    void move(std::list<Layer*>& s, std::list<Layer*>& e, int end);//移动塔
    void showTower();//显示塔
    void clearTower();//清空塔元素
    void sloveTower(int level, int src, int medium, int dest);//自动求解塔
    void clearStep();//清空解空间
    void saveFileTower(int level, int timeShift);//存档塔
    void readFileTower(int& level, int& timeShift);//读档初始化塔
};
```

图 3.11 HanoiTower 类的成员

HanoiTower 类的主要数据成员是汉诺塔。汉诺塔的三根柱子用三个顺序容器（A，B，C）表示，汉诺塔每层的“块”用 Layer 类的对象表示。

建立新游戏时，会初始化汉诺塔。初始化汉诺塔时，会从大到小生成若干个 Layer 类的对象，并使用头插法将其指针插入 A 中（与栈类似）。移动汉诺塔时，若终点汉诺塔没有元素或起点汉诺塔顶层宽度小于终点汉诺塔顶层宽度，系统会先修改起点塔顶指针指向的 Layer 对象中的位置坐标，之后再把起点塔顶的元素（指针）弹出并将其插入终点塔顶。当塔 C 中的“块”数目与初始化“块”的数目相同时，游戏胜利。在建立新游戏时和返回主菜单时，会将“块”所占的空间释放并清空塔中元素。

移动塔上的“块”时调用 moveTower()即可，该函数的形参为起点标号和终点标号，不必关心塔移动的具体过程。类中有 startTower 和 endTower 两个参数，存储了移动塔的起点和终点。三个 MoveTowerButton 类按键可修改这两个参数，之后调用 moveTower()即可移动塔上的“块”。

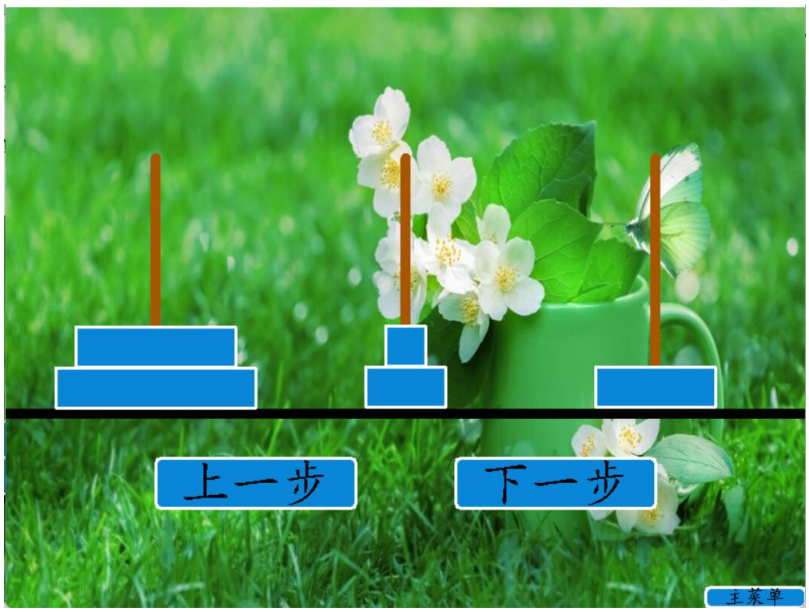

图 3.12 游戏教程界面

在游戏教程功能中，需要演示正确的汉诺塔移动步骤，这里用二维数组 step[64][2]存储所有步骤，每行是一个步骤，每个步骤包含一个起点标号和一个终点标号。此游戏最高难度为 6，递归法需要 63 步可解，因此该数组大小合适。当用户点击教程时，系统会根据用户选择的难度用递归法生成解，存入数组。类中用一个下标指针 nowStep 指示当前是哪一步（行），当用户点击下一步时，就用指针处元素执行一次 moveTower()，再将指针后移，当用户点击上一步时，就将指针前移，再将指针处起点与终点交换（传参时交换，不改变数组元素）传入 moveTower()，执行 moveTower()。

游戏有存档功能，可将汉诺塔相关信息存入文件中，以便下次读取。saveFileTower()可将汉诺塔数据存入文件 TowerLoad.txt。数据格式如图 3.13。其中 timeShift 的含义会在之后解释。读取存档时会用这些变量建立汉诺塔，修改游戏参数，达到读档的效果。

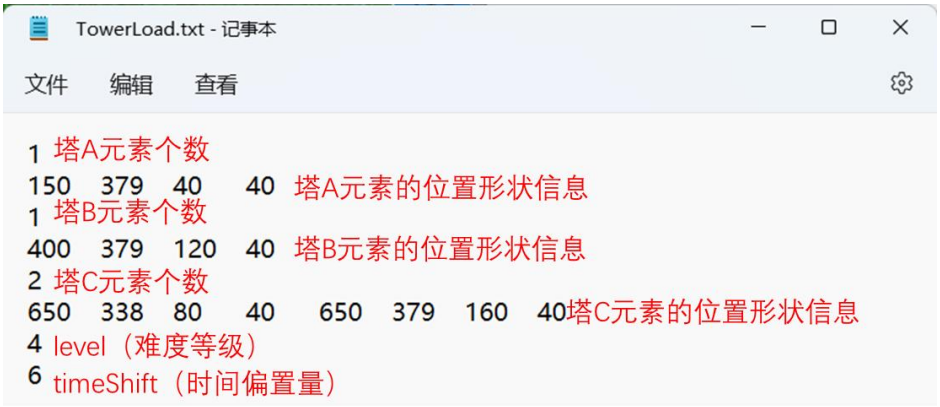

图 3.13 汉诺塔存档数据格式

类中的 stepNum 成员和 stepShift 成员是记录玩家在游戏中的移动步数的，只有符合游戏规则的移动才会被记录。stepShift 是移动步数偏移值，记录的是上次存档的移动步数，新游戏的 steppShift 为 0，总移动步数=stepNum+stepShift。

3.8 MyClock 类

MyClock 类是一个时钟，它的主要功能就是记录游戏时间。使用时将其当作一个秒表即可。该类主要使用了标准库<chrono>，内部有两个 time_point 对象，分别存储秒表的起点和终点。

MyClock 类有两个外部接口，一个是 resetClock()，作用是将当前时间置为“时间零点”，另一个是 getClock()，返回当前时间到“时间零点”的秒数

```
using namespace std::chrono;
class MyClock
{
protected:
    steady_clock::time_point start, end;//记录时间的起点和终点
public:
    void resetClock();  //开始定时（将当前时刻置为0）
    int getClock();     //获取当前定时时间
};
```

图 3.14 MyClock 类

3.9 Rank 类

Rank 类用来记录游戏排行榜，也可读写文件。

Rank 类中有数据成员 rankList，用固定大小数组存储 6 个结构体 Score，一个 Score 结构体中存储了三个字符串，某难度下的游戏记录（前三名），每个字符串存储的是秒数。有一相同大小的数组 rankTime，其中存储的是“x 分 x 秒”字符串。可通过 score2time() 将 rankList 中的数据转换到 rankTime 中。

Rank 类可将排行榜数据存储到本地文件 Rank.txt 中，游戏启动时也会读取本地文件中的排行榜数据。数据格式如图 3.16，从第 1 行到第 6 行分别是难度 1 到难度 6 的前三名的成绩数据，-1 表示无记录。图 3.17 是排行榜在游戏中的显示效果。

Rank 类中 newRank()可将新纪录插入排行榜，并输出提示框。提示框如图 3.18。

```
typedef struct
{
    std::string first;
    std::string second;
    std::string third;
}Score;
class Rank
{
protected:
    std::array<Score, 6> rankList;//6个难度等级，存储秒数
    std::array<Score, 6> rankTime;//6个难度等级，x分x秒字符
    std::string fileName;
    void  score2time();//秒数转换为x秒x分
public:
    Rank();
    void ReadRankFile();//读文件
    void SaveRankFile();//写文件
    void showRank();//显示排行榜
    void newRank(HWND hnd, int level, int time);//新纪录
};
```

图 3.15 Rank 类

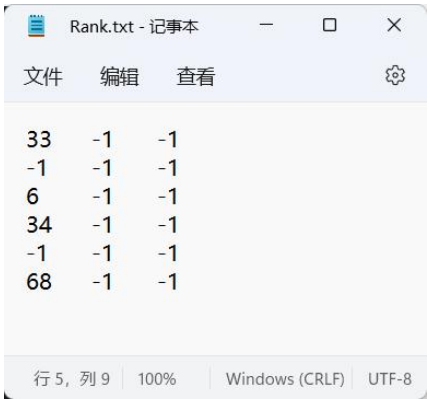

图 3.16 Rank.txt 数据格式

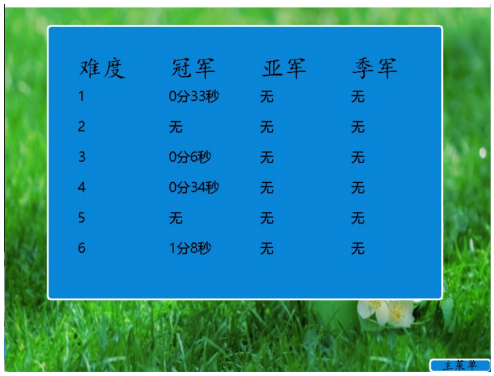

图 3.17 游戏中的排行榜

图 3.18 破纪录提示框

3.10 Record 类

Record 用于记录玩家的游戏记录，也具有显示功能。

Record 类的的主要成员是一个存储 RecordData 指针的 vector 容器 RecordList。

RecordData 结构体有 4 个数据成员，分别为游戏等级 level，步数 step，花费时间 time，是否成功 isSuccess。每次增加记录都会申请一个 RecordData 类型的新空间，之后对新空间内的数据进行赋值，最后将其指针插入 RecordList 的后面。

```
typedef struct
{//数据类型
    int level;
    int step;
    int time;
    int isSuccess;
}RecordData;
class Record
{
protected:
    std::vector<RecordData*> RecordList;
    int beginRecordPage = 0;//指示当前起始页码
public:
    void addRecord(int level, int step, int time, int isSuccess);
    void showRecord();//显示记录
    void saveRecord();//将记录存入文件
    void readRecord();//读取文件中的记录
    void addRecordPage();//页码增加
    void subRecordPage();//页码减小
    void startRecordPage();//首页
    void endRecordPage();//尾页
};
```

图 3.19 Record 类

beginRecordPage 指示了一个起点下标，显示函数会显示 RecordList 中从此下标开始的 10 个元素。可通过 Game 类中的按键控件控制 beginRecordPage 的值，达到翻页效果，共有四个翻页按键，分别为“上一页”、“下一页”、“首页”、“尾页”，这些按键事件对应函数 subRecordPage()、addRecordPage()、startRecordPage()、endRecordPage()。

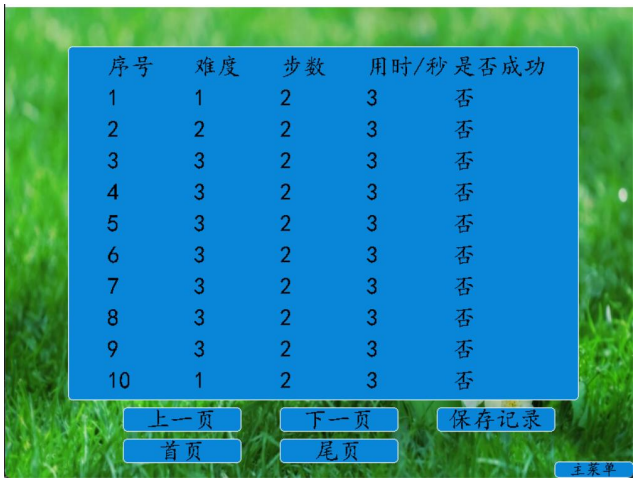

图 3.20 Record 类显示效果

saveRecord()可将记录写入文件 Record.txt，readRecord()可从 Record.txt 中读取记录。数据格式如图 3.21。每行的四个数字依次为：难度，步数，用时/秒，是否成功。

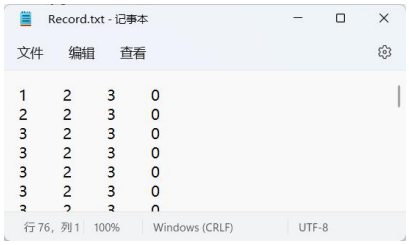

图 3.21 Record.txt

3.11 Game 类

在游戏架构设计章节中已将 Game 类的基本情况介绍完毕，这里补充一些细节。

3.11.1 游戏时间计算

游戏中计算时间时会在当前时间上加上一个时间偏置值 timeShift，该变量主要是为了读取游戏存档。新游戏的 timeShift 为 0，读档游戏的 timeShift 为文件中的值。当时间超过上限就宣布游戏失败，游戏各难度的时间上限如表所示。

表 3.1 各难度的时间上限

难度	1	2	3	4	5	6
时间上限	10 秒	30 秒	1 分	4 分	6 分	10 分

3.11.2 图像显示及按键事件实现

以音乐开关按键为例，介绍图像显示过程和按键事件触发过程。

（1）在 Game 类中定义音乐开关控件 VolumeButton 及相关变量 Volume。Volume 表示当前音乐的开关状态。

```
bool Volume = true;//默认声音开
PushButton VolumeButton;//音乐按键
```

图 3.22 音乐按键定义

（2）在合适的地方初始化控件，这里在 Game 类构造函数的列表中初始化。

```
Game::Game()
    :menu_text_hnt("汉诺塔", 100, 250, 60), VolumeButton("关闭音乐", 0, 579, 100, 20, 20)
    , level_text_hnt("选择难度", 80, 250, 50), MenuButton("主菜单", 700, 579, 100, 20, 20)
```

图 3.23 音乐按键初始化

（3）将该控件的绘图函数放进 Game 类的 GameDraw()中。音乐按键应该出现在主菜单中，故绘图函数应放在 menu 系统状态下。此时运行程序可在主菜单看到音乐按键。

```
void Game::GameDraw()
{
    //绘制背景
    putimage(0, 0, &bkImage);
    //显示主菜单按键
    MenuButton.show();
    switch (op)//根据系统的状态显示不同的界面
    {
    case menu:
        //绘制标题
        menu_text_hnt.showText();
        //绘制音乐按键
        VolumeButton.show();
        for (auto btn : menu_btns)
        {
            btn->show();
        }
        break;
    case tutorial://教程与新游戏界面1相同
```

图 3.24 音乐按键显示

（4）这里为按键增加交互功能，程序应写在 Game 类的 undateWithInput()中，这里也要注意系统状态，应放在 menu 系统状态下。

系统会不断地获取鼠标键盘输入消息，若没有消息则什么都不干，若有消息，就将消息传入控件，控件会根据消息内容（主要是鼠标的坐标）执行对应操作。

```
void Game::undateWithInput()
{
    // 获取一条鼠标或按键消息
    ExMessage msg;
    //若消息存在，则执行对应行为，否则什么也不干
    if (peekmessage(&msg, EX_MOUSE | EX_KEY))
    {
        //主菜单按键
        MenuButton.eventLoop(msg);
        if (MenuButton.isClicked()) { MainMenu(); }
        //根据系统状态执行不同操作
        switch (op)
        {
        case menu:
```

图 3.25 undateWithInput()获取消息

我们将图 3.26 中的音乐开关控制程序写入图 3.25 中的 case menu:中即可实现音乐开

关控制。首先调用方法 eventLoop()将消息传入控件，再用方法 isClicked()判断按键是否被点击（其实是判断鼠标是否在按键显示的区域内执行了点击操作），若被点击则执行对应操作，否则什么也不干。

```
VolumeButton.eventLoop(msg);
if (VolumeButton.isClicked())
{//音量开关

    Volume = !Volume;//音乐状态取反
    if (Volume)//当前音乐状态应为开
    {
        VolumeButton.setText("关闭音乐");
        mciSendString("open ./sources/bkMusic.mp3 alias BGM", 0, 0, 0);
        mciSendString("play BGM repeat", 0, 0, 0);//重复播放
    }
    else//当前音乐状态应为关
    {
        VolumeButton.setText("打开音乐");
        mciSendString("close BGM", 0, 0, 0);//关闭音乐
    }
}
break;
```

图 3.26 音乐按键点击事件

（5）若控件有需要被更新的参数且此更新与输入无关，则应写入 Game 类的 undateWithoutInput()中。音乐按键没有相关功能。

3.11.3 游戏结束

这里介绍游戏是如何结束的。游戏失败和胜利的程序放在 Game 类的 undateWithoutInput()中，表示游戏结束，之后释放申请的空间（Layer 对象），清空相关数组，返回主菜单。

在游戏结束函数中会增加一个游戏记录，并将游戏记录保存到 Record.txt 中，不论游戏是否失败。

```
void Game::gameover(int gameOverState/*=1*/)
{
    //增加，保存游戏记录
    addRecord(level, stepNum + stepShift, finalTime, gameOverState);
    saveRecord();
```

图 3.27 增加一个游戏记录并保存

游戏失败的条件是是剩余时间为 0，游戏开始时会重置“时间零点”，在游戏过程中系统会不断地计算当前花费了多少时间，公式为 timeShift+getClock()，timeShift 为时间偏置值，新游戏的 timeShift 为 0，读档游戏的 timeShift 为文件中的值，getClock()可获取当前距“时间零点”的时间长度（单位：秒）。当用户花费时间大于等于游戏时间上限时，则游戏结束，弹出提示框。

游戏胜利的条件是塔 C 中的元素个数与难度相等（汉诺塔全部被移动到塔 C 上），此时会弹出游戏胜利提示框，更新游戏记录，返回主菜单。

游戏教学结束的条件塔 C 中的元素个数与难度相等（汉诺塔全部被移动到塔 C 上），此时会弹出教程结束提示框，将为 Layer 对象申请的空间释放，清空 step[][]中的步骤，重置 nowStep，step，stepNum，stepShift 等相关变量。

```
void Game::undateWithoutInput()
{
    switch (op)//根据系统状态执行不同操作
    {
    case readFile://读档
    case newGame_Game:
        finalTime = timeShift + getClock();//更新当前花费时间
        if (C.size() == level)//游戏胜利
            gameover();
        switch (level)
        {//游戏失败
        case 1:if (10 - finalTime <= 0) gameover(0); break;
        case 2:if (30 - finalTime <= 0) gameover(0); break;
        case 3:if (60 - finalTime <= 0) gameover(0); break;
        case 4:if (240 - finalTime <= 0) gameover(0); break;
        case 5:if (360 - finalTime <= 0) gameover(0); break;
        case 6:if (600 - finalTime <= 0) gameover(0); break;
        }
        break;
    case tutorial_Game://游戏教学模式
        if (C.size() == level)//演示结束
            tutorialover();
        break;
    }
}
```

图 3.28 游戏结束

图 3.29 游戏胜利

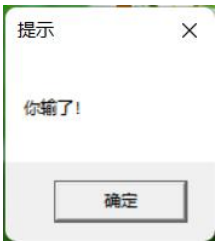

图 3.30 游戏失败

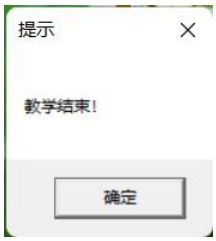

图 3.31 游戏教学结束

参考文献

[1] 童晶，丁海军，金永霞，周小芹编著. 面向“工程教育认证”计算机系列课程规划教材 C 语言课程设计与游戏开发实践教程. 北京：清华大学出版社, 2017.07.

[2] Stanley B. Lippman，Josee Lajoie，Barbara E. Moo 著. C++ Primer 中文版. 北京：电子工业出版社, 2013.09.

[3] 郑莉，董渊编著. C++语言程序设计 第 5 版. 北京：清华大学出版社,