【数据结构与算法综合实验】欢乐连连看(C++ & MFC)案例

原创

拾年之璐 于 2019-04-26 23:08:39 发布 7106 收藏 62
分类专栏： 本科课程笔记 文章标签： 武汉理工大学 欢乐连连看 MFC 数据结构与算法

本文链接：https://blog.csdn.net/cxh_1231/article/details/89577820


数据结构实验代码集 同时被 3 个专栏收录
8 篇文章 5 订阅
订阅专栏


数据结构&算法笔记
13 篇文章 3 订阅
订阅专栏


本科课程笔记
32 篇文章 11 订阅
订阅专栏

- 说明：这是武汉理工大学计算机学院数据结构与算法综合实验课程的第三次项目：欢乐连连看(C++ & MFC)迭代开发代码。
- **>>点击查看武汉理工大学计算机专业课程资料汇总**
- >>点击查看WUTer计算机专业实验汇总
- **谨记：纸上得来终觉浅，绝知此事要躬行。**

注意：

本文所有内容已经废弃。

新文章请移步：https://blog.csdn.net/cxh_1231/article/details/109717884

笔者本科资料汇总：https://blog.csdn.net/cxh_1231/article/details/112465790

写在文章开始：

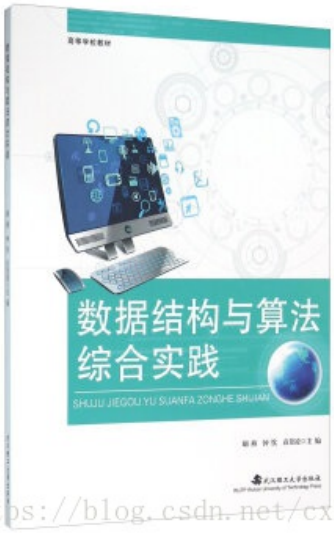
我想，能搜到这篇博客的同学，应该是WUTer吧。

既然你看到了这篇博客，就请听学长啰嗦几句。。。

关键【数据结构与算法综合实验】的这三次实验来说，虽然总体难度较大，但是老师都给了视频的，把视频看完，按部就班的来做，基本上都能够完成最终的实验课。这不是重点，重点是这次实验要有所收获。其实代码到处都能找到，找你们的学长学姐，他们都有。找不到学长学姐，这不是有学长在网上给你们分享嘛（^_^）。不可否认，我当时做实验的时候，也是参考学长的代码（虽然他的全是BUG），然后结合视频来做的。重要的是理解这些代码在干什么，各个算法，代码之间的逻辑性，以及代码之间的调用关系，这些在以后的项目中都会用到（感兴趣可以去搜索一下**MVC模型**，这次实验基本是这种模式）。

总之，所有的一切要亲自动手试试，正如我前面说的：**纸上得来终觉浅，绝知此事要躬行。**只有动手试试之后，你才会发现，理论和现实之间有很大的差距。。。

实验教材：



任务列表：

1. 创建工程
2. 绘制欢迎语
3. 主界面设计
4. 游戏界面设计
5. 绘制游戏地图
6. 同色消子
7. 程序结构调整
8. 消子判断
9. 数据结构设计
10. 开始游戏
11. 消子
12. 判断胜负
13. 提示
14. 重排
15. 计时
16. 帮助
17. ……

实验目的以及要求：

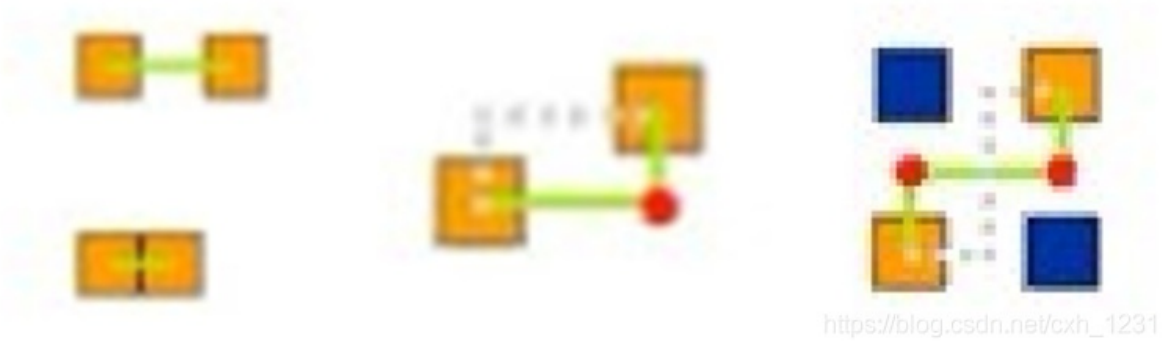
1.目的

1. *调研连连看游戏，了解连连看游戏的功能和规则等。*
2. *掌握集成开发工具（Visual C++ 6.0 或 Microsoft Visual Studio 2010）。*

3. *掌握 C++的基础编程。*
4. *了解 MFC 框架，包括 MFC Dialog 应用程序和 GDI 编程。*
5. *了解线性结构，重点掌握数组和栈操作，掌握数组的遍历、消子胜负判断等算法。*
6. *进一步掌握图的数据结构和应用，重点了解邻接矩阵的存储结构。*
7. *掌握图的常用算法，比如，深度优先遍历、图的联通判断等，能够利用图的算法实现游戏联通判断和胜负判断。*
8. *了解企业软件开发过程，了解系统需求分析和设计，应用迭代开发思路进行项目开发。*
9. *养成良好的编程习惯和培养软件工程化思维，综合应用“C++编程、MFC Dialog、算法、线性结构或者图”等知识，开发“连连看游戏”桌面应用程序，达到掌握和应用线性结构以及非线性结构核心知识的目的。*

2.要求

1. *连连看游戏是对一堆图案中的相同图案进行配对消除的简单游戏，在一定的规则之内对相同的图案进行消除处理，在规定的时间内消除所有的图案后玩家获胜。*
2. *利用MFC Dialog应用程序设计游戏主界面，通过鼠标点击事件实现游戏界面加载；*
3. *通过图的相关知识，完成“开始游戏”操作，进一步实现“消子”“判断胜负”“提示”“重排”“计时”等功能。*
4. *游戏规则是一条直线消子、两条直线消子、三条直线消子。*



实现说明：

在Microsoft Visual Studio 2017集成开发环境中新建一个MFC对话框（Dialog）工程，解决方案：Lianliankan，项目名称：欢乐连连看，工程名称：LLK，主对话框：CLLKDlg，然后修改主界面的对话框属性值，如窗口标题、窗体按钮、对话框图标等。

CLLKDlg类为主界面对话框类，在该类中添加游戏界面“基本模式”按钮BN_CLICKED消息响应函数CLLKDlg：：OnBnClickedBtnBasic()。创建并显示游戏界面对话框，并控制主界面的隐藏和显示。

为项目添加CGameDlg对话框类，该类为游戏界面，与用户进行信息交换，为其添加相关数据成员和成员函数，相关数据成员和成员函数实现的功能如下表：

数据成员	描述
CDC m_dcMem	位图内存DC，游戏界面的背景图片保存到该DC中
CDC m_dcElement	元素内存DC
CDC m_dcMask	掩码内存DC
CGameControl m_GameC	游戏控制类对象

成员函数	描述
void InitBackground();	加载游戏背景图片到位图DC中，设置游戏界面对话框的大小
virtual BOOL OnInitDialog();	对话框初始化函数，并调用InitBackground()

成员函数	描述
afx_msg void OnPaint();	对话框WM_PAINT消息响应函数，将游戏背景图片绘制到对话框界面中
void InitElement();	初始化游戏元素和掩码
void UpdateMap();	更新游戏地图
void OnClickedButtonStart();	“开始游戏”按钮BN_CLICKEN消息响应函数，开始游戏，生成游戏地图，并显示游戏地图
void UpdateWindow();	更新界面，从生成的游戏地图图结构中取出顶点，根据顶点从元素图片中取出相对应的元素，显示在游戏地图中
void DrawTipFrame(int, int);	绘制游戏提示框
void DrawTipLine(Vertex,int)	绘制提示线
void OnClickedButtonStop();	“暂停游戏”按钮BN_CLICKEN消息响应函数
void OnClickedButtonTips();	“提示”按钮BN_CLICKEN消息响应函数
void OnClickedButtonRestart();	“重排”按钮BN_CLICKEN消息响应函数
void OnBnClickedButtonHelp();	“帮助”按钮BN_CLICKEN消息响应函数
void OnBnClickedButtonSet();	“设置”按钮BN_CLICKEN消息响应函数

为工程项目添加CGameControl类，该类为游戏的控制类，实现整个项目的控制，其相关数据成员和成员函数的实现描述如下表：

数据成员	描述	
CGraph m_graph;	保存生成的游戏地图的图结构	
Vertex m_ptSelFirst;	记录第一次选中的点信息	
Vertex m_ptSelSec;	记录第二次选中的点信息	

成员函数	描述
void StartGame(void);	开始游戏，调用CGameLogic类的成员函数，生成游戏地图的图结构
int GetElement(int nRow, int nCol);	从游戏地图的图结构中取出顶点，用于界面第nRow行、nCol列显示元素的数据
void SetFirstPoint(int nRow, int nCol);	设置第一个点
void SetSecPoint(int nRow, int nCol);	设置第二个点
bool Link(Vertex avPath[MAX_VERTEX_NUM], int &nVexnum);	判断是否连通

数据成员	描述	
bool IsWin(int nTime);	判断是否在规定时间内消子完成而获胜	
void Reset(void);	实现图的重排	

为工程添加游戏业务逻辑CGameLogic类，实现逻辑上的相关操作，其相关数据类型和成员函数的实现描述如下表：

数据成员	描述
Vertex m_avPath[MAX_VERTEX_NUM];	保存连接路径的起始点、拐点、终点
int m_anPath[MAX_VERTEX_NUM];	保存在进行连接判断时所经过的顶点
int m_nCorner;	保存路径中的拐点数
int m_nVexNum;	表示顶点数

成员函数	描述
void InitMap(CGraph &graph);	根据随机生成的数据，初始化游戏地图
void UpdateArc(CGraph &graph, int nRow, int nCol);	判断游戏地图中的nRow行、nCol列的顶点与它上下左右的顶点是否有边，如果有边，则更新图结构
bool IsLink(CGraph &graph, Vertex v1, Vertex v2);	判断v1、v2两个点是否连通
void Clear(CGraph &graph, Vertex v1, Vertex v2);	将v1、v2两个点进行消子
bool SearchPath(CGraph &graph, int nV0, int nV1);	使用深度优先搜索算法判断两个点是否连通
bool IsExsit(int nVi);	判断顶点是否已在路径中存在
bool IsCornor(void);	判断拐点的有效性
void PushVertex(int v)/ PopVertex();	添加/取出一个路径顶点
int GetVexPath(Vertex avPath[MAX_NUM]);	得到路径，返回的是顶点数
bool IsBlank(CGraph &graph);	判断图中顶点是不是全是空
bool SearchValidPath(CGraph &graph);	寻找可以进行消除的图片，用于提示
void ResetGraph(CGraph& graph);	实现图结构的重排

为工程添加程序的数据结构类CGraph类，从图的结构上实现消子等相关功能，其其相关数据类型和成员函数的实现描述如下表：

数据成员	描述
Vertices m_Vertices;	一维数组，保存连连看游戏地图中的顶点

数据成员	描述
AdjMatrix m_AdjMatrix;	二维数组，保存连连看游戏中的顶点的边
int m_nVexnum;	顶点个数
int m_nArcnum;	边数量

成员函数	描述
void InitGraph();	初始化图结构
int AddVertex(int nInfo);	添加顶点，并获得顶点个数
void AddArc(int nV1Index,int nV2Index);	添加生成边
int GetVertex(int nIndex);	获取顶点索引号
bool GetArc(int nV1Index, int nV2Index);	获得两个顶点的边信息
void UpdateVertex(int nIndex,int info);	更新顶点信息，将图顶点数组中索引号为nIndex的顶点的值更新为info
int GetVexnum();	获取图中顶点的个数
void ClearGraph(void);	清理图结构
void ChangeVerex(int nIndex1, int nIndex2);	调换两个点的位置

为项目添加全局头文件global.h，用来存放整个项目需要的数据，相关数据设计如下表：

#define BLANK	-1	
#define MAX_ROW	10	//初始行数
#define MAX_COL	12	//初始列数
#define MAX_VERTEX_NUM	120	//顶点数
#define MAX_PIC_NUM	10	//图片花色
#define REPEAT_NUM	12	//每种花色图片个数
#define MAP_TOP	50	//游戏地图左上角纵坐标
#define MAP_LETF	50	//游戏地图左上角横坐标
#define PIC_HEIGHT	40	//游戏地图高度
#define PIC_WIDTH	40	//游戏地图宽度
#define PLAY_TIMER_ID	1	//计时器的编号
#define GAME_LOSE	-1	//失败
#define GAME_SUCCESS	0	//获胜

#define GAME_PLAY	1	//游戏正在进行

项目代码：

由于整个项目代码非常多，所以以下只展示关键的代码。

完整代码请点击此处前往下载！

初始化游戏地图算法：随机生成打乱地图代码：

```
//初始化游戏地图
void CGameLogic::InitMap(CGraph &graph){
 //随机生成地图
 int anTemp[MAX_VERTEX_NUM];
 //多少花色
 for (int i = 0; i < MAX_PIC_NUM; i++) {
  //重复数
  for (int j = 0; j < REPEAT_NUM; j++)
  {
   anTemp[i * REPEAT_NUM + j] = i;
  }
 }
 srand((int)time(NULL)); //设置种子
 //随机任意交换两个数字
 for (int i = 0; i < 300; i++) {
  //随机得到两个坐标
  int nIndex1 = rand() % MAX_VERTEX_NUM;
  int nIndex2 = rand() % MAX_VERTEX_NUM;
  //交换两个数值
  int nTemp = anTemp[nIndex1];
  anTemp[nIndex1] = anTemp[nIndex2];
  anTemp[nIndex2] = nTemp;
 }
 //初始化顶点
 for (int i = 0; i < MAX_VERTEX_NUM; i++) {
  graph.AddVertex(anTemp[i]);
 }
 //更新弧信息
 for (int i = 0; i < MAX_ROW; i++) {
  for (int j = 0; j < MAX_COL; j++)  {
   UpdateArc(graph, i, j);
  }
 }
}
```

连通判断算法：使用深度优先搜索算法实现

```
//连接判断函数
bool CGameLogic::IsLink(CGraph &graph, Vertex v1, Vertex v2){
 //获得顶点索引号
 int nV1Index = v1.row * MAX_COL + v1.col;
 int nV2Index = v2.row * MAX_COL + v2.col;
 PushVertex(nV1Index);  //压入第一个点
 //搜寻两点之间的连通路径
 if (SearchPath(graph, nV1Index, nV2Index) == true) {
  return true;
 }
 PopVertex();
 if (v1.row == v2.row) {
  if (v1.row == 0 || v1.row == MAX_ROW - 1) {
   return true;
  }
 }
 if (v1.col == v2.col) {
  if (v1.col == 0 || v1.col == MAX_COL - 1) {
   return true;
  }
 }
 return false;
}
//使用深度优先搜索法搜寻一条有效连通路径
bool CGameLogic::SearchPath(CGraph &graph, int nV0, int nV1){
 int nVexnum = graph.GetVexnum();//得到顶点数
 //遍历图中nV0行，从0列到nVexnum列，值为true的点
 for (int nVi = 0; nVi < nVexnum; nVi++) {
  if (graph.GetArc(nV0, nVi) && !IsExsit(nVi)){
   //压入当前顶点，假设为路径的一个有效顶点
   PushVertex(nVi);
   //当拐点数大于2 时，直接放弃该顶点
   if (m_nCorner > 2){
    PopVertex();       //取出压入的顶点，与PushWertex(nVi)对应
    continue;
   }
   //当前顶点不是nVi时，继续搜寻下一个相邻且连通的顶点
   if (nVi != nV1)   {
    //当中间顶点不为空时，表示该条路径不通
    if (graph.GetVertex(nVi) != BLANK)    {
     PopVertex();    //取出压入的顶点，与PushWertex(nVi)对应
     continue;
    }
    //如果nVi是一个已消除的点，则判断（nVi，nV1）是否连通
    if (SearchPath(graph, nVi, nV1)){
     return true;
    }
   }
   else{
    return true;
   }
   PopVertex();     //取出压入的顶点，与PushWertex(nVi)对应
  }
 }
 return false;
}
```

提示功能算法：

```
//提示按钮功能实现
bool CGameControl::Help(Vertex avPath[MAX_VERTEX_NUM], int & nVexnum)
{
 CGameLogic logic;

 //判断是否为空
 if (logic.IsBlank(m_graph) == true)
 {
  return false;
 }
 //查找一个有效的提示路径
 if (logic.SearchValidPath(m_graph))
 {
  //返回路径顶点
  nVexnum = logic.GetVexPath(avPath);

  return true;
 }
 return false;

}
//提示算法
bool CGameLogic::SearchValidPath(CGraph& graph){
 //得到顶点数
 int nVexnum = graph.GetVexnum();
 for (int i = 0; i < nVexnum; i++){
  //得到第一个非空顶点
  if (graph.GetVertex(i) == BLANK){
   continue;
  }
  //遍历得到第二个同色顶点
  for (int j = 0; j < nVexnum; j++){
   if (i != j)
   {
    //如果第i个点和第j个点同色
    if (graph.GetVertex(i) == graph.GetVertex(j)){
     //压入第一个点
     PushVertex(i);
     if (SearchPath(graph, i, j) == true){
      return true;
     }
     //取出压入的顶点时，与PushVertex(i);对应
     PopVertex();
    }

   }
  }
 }
 return false;
}
```

重排功能算法：根据随机数交换两张图片位置

```
//实现图结构的重排
void CGameLogic::ResetGraph(CGraph& graph){
 //随机交换顶点数组中两个顶点的值
 for (int i = 0; i < 200; i++){
  //随机得到两个坐标
  int nIndex1 = rand() % MAX_VERTEX_NUM;
  int nIndex2 = rand() % MAX_VERTEX_NUM;
  //交换两个数值
  graph.ChangeVerex(nIndex1, nIndex2);
 }
 //更新弧信息
 for (int i = 0; i < MAX_ROW; i++){
  for (int j = 0; j < MAX_COL; j++){
   UpdateArc(graph, i, j);
  }
 }
}
```

运行结果截图：

- **游戏主界面：**

- **游戏界面：**

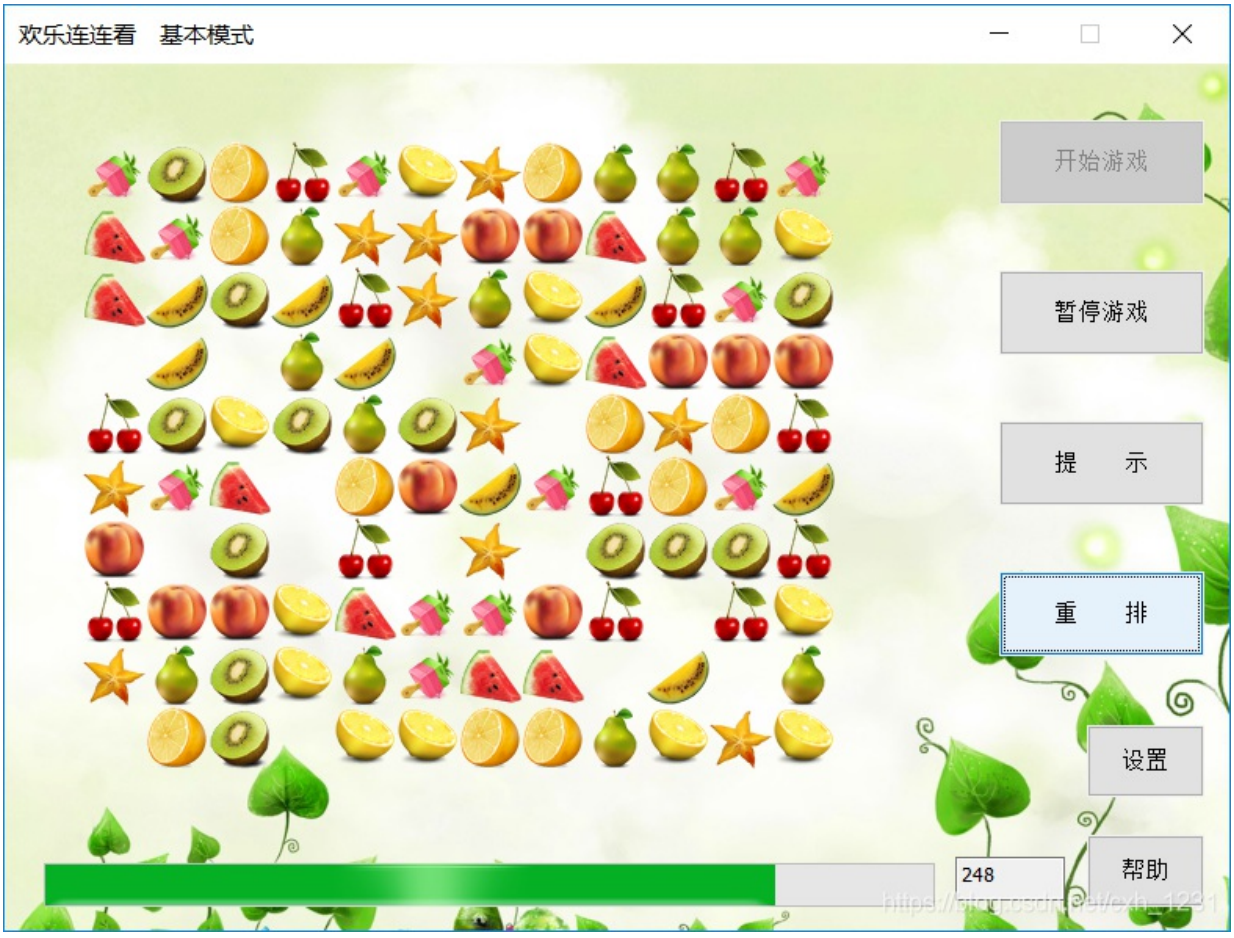

欢乐连连看　基本模式
开始游戏
暂停游戏
提　示
重　排
设置
帮助
271

- 暂停界面


欢乐连连看　基本模式
开始游戏
继续游戏
提　示
重　排
设置
帮助
265

- 连线界面

- 上图点击重排后界面

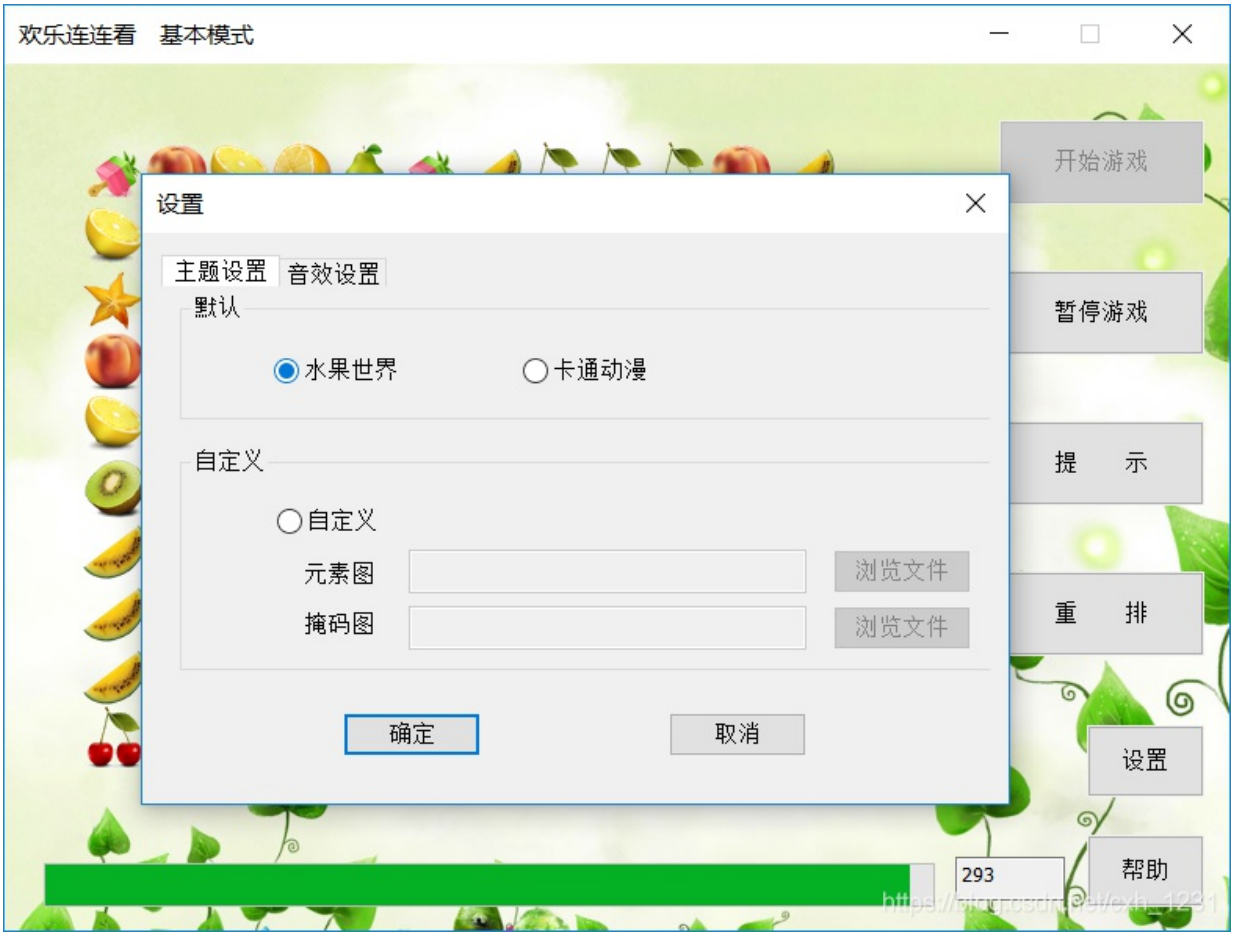


- 两个拐角连线界面：

- **完成游戏获胜界面：**

- **点击“帮助”按钮界面：**

- 点击“设置”按钮界面：



The End!