任务 7 操作步骤

1.1 判断胜负

1.1.1 功能需求

1、介绍

在基本模式下如果将游戏地图中所有的图片都消除，则提示玩家获胜，并且可以重新开始新游戏。

2、输入

游戏地图中剩余图片数量。

3、处理

判断游戏地图中所有的图片是否都被消除。

4、输出

(1) 如果游戏地图中所有图片都消除了，则提示玩家获胜，并且可以通过开始游戏按钮重新开始新游戏。

(2) 如果没有都消除，玩家不能开始新游戏。直到所有图片都消除完。

1.1.2 设计思路

在“消子判断”的基础上进行迭代开发。

1、判断胜负的条件

界面上游戏地图中所有的图片都被消除。

2、判断胜负的功能实现

每次成功消除一对图片后，在 CGameLogic 中，判断 CGameControl 类保存游戏地图图结构 m_graph 中，顶点数组所有元素的值是否为空(-1)。为空表示该顶点已经被消除。

3、类设计

(1) CGameLogic 类

成员函数	描述
bool IsBlank(int** pGameMap)	判断图中顶点是不是全为空

(2) CGameControl 类

成员函数	描述
bool IsWin(void)	判断是否获胜

(3) CGameDlg 类

数据成员	描述
bool m_bPlaying	游戏状态标识

1.1.3 编码实现

导入“消子判断”的解决方案，在此基础上进行迭代开发，实现步骤如下：

步骤一：判断胜负。

步骤二：控制开始游戏按钮状态。

1、判断胜负

判断胜负的规则为：游戏地图中所有的图片都被消除完，也就是图结构中的顶点数组中所有元素都为空(-1)。

(1) 逻辑层判断胜负

在 CGameLogic 类中，添加 IsBlank()函数。判断图中所有的顶点是否都为空。

IsBlank()函数定义为：bool IsBlank(int** pGameMap)。

(2) 控制层判断胜负

在 CGameControl 类中，添加 IsWin()函数，调用 CGameLogic::IsBlank()函数，判断是否已经获胜。IsWin()函数定义为：bool IsWin(void)。

(3) 界面层判断胜负

在 CGameDlg::OnLButtonUp()函数中，调用 CGameControl::IsWin()函数，判断是否已经获胜。如果获胜，则提示用户获胜。

```
void CGameDlg::OnLButtonUp(UINT nFlags, CPoint point)
{
    //……
    // 判断胜负
    if(bSuc && m_GameC.IsWin())
    {
        //……
    }
    //……
}
```

2、控制开始游戏按钮状态

游戏开始后，到游戏获胜前，开始按钮的状态为禁用。直到游戏获胜之后，才可用。

设置按钮的禁用状态，可以调 CWnd::EnableWindow()函数，参数为 FALSE 表示该对象不可用。参数为 TRUE 表示该对象可用。

(1) 给 CGameDlg 类添加数据成员 bool m_bPlaying，表示游戏是否正在进行。

(2) 在构造函数中初始化 m_bPlaying = false。

(3) 在 CGameDlg::OnBnClickedBtnStart()函数中，开始游戏时，将 m_bPlaying 赋值为 true。并将开始游戏按钮设置为禁用状态。

(4) 在 CGameDlg::OnLButtonUp()函数中，首先判断游戏是否正在进行，如果没有正在进行，则不响应鼠标事件。然后在游戏获胜后将 m_bPlaying 赋值为 false，最后将设置开始游戏按钮的状态为可用。

(5) 编译并运行程序。