# 任务 5 操作步骤

## 1.1 创建工程

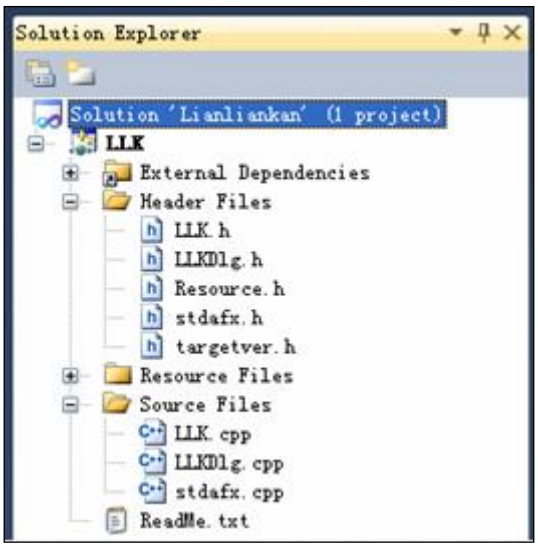
### 1.1.1 功能需求

创建“欢乐连连看”游戏工程。


**图**错误!文档中没有指定样式的文字。**-1 游戏主窗口**

**1、创建工程**

(1) 开发工具：Microsoft Visual Studio 2010
(2) 解决方案：Lianliankan
(3) 工程类型：MFC 对话框(Dialog)工程
(4) 项目名称：欢乐连连看
(5) 工程名称：LLK
(6) 主对话框：CLLKDlg




**图**错误!文档中没有指定样式的文字。**-2 解决方案目录**

**2、游戏主窗口**

(1) 窗口标题：欢乐连连看。

(2) 标题栏按钮：最小化按钮、关闭按钮。最大化与还原按钮不可使用。

(3) 对话框图标：自定义图标，ico 文件。


**图**错误!文档中没有指定样式的文字。**-3 程序图标**

## 1.1.2 设计思路

选用 Miscrosoft Visual Studio 2010 作为开发工具，简称 VS2010。选择 MFC 对话框工程作为游戏工程。

## 1.1.3 编码实现

**实现步骤如下：**

步骤一：创建解决方案。

步骤二：创建工程。

步骤三：修改主界面对话框属性。

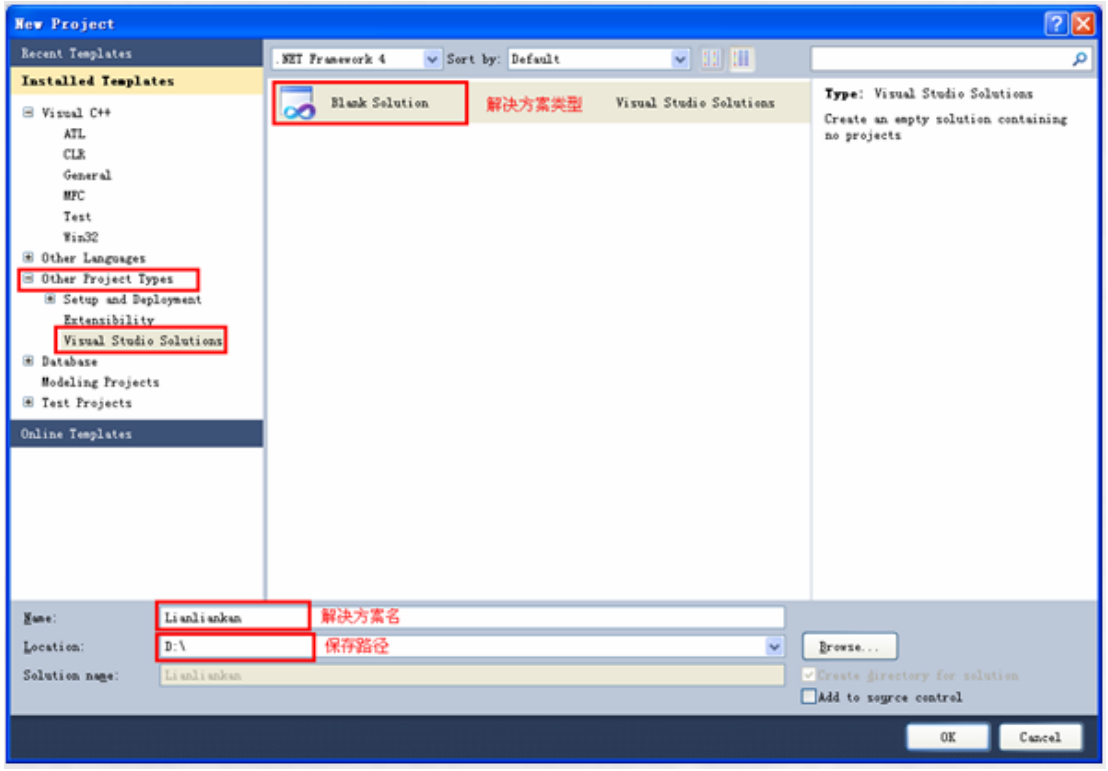
**1、创建解决方案**

(1) 选择“开始 -> 程序 -> Miscrosoft Visual Studio 2010 -> Miscrosoft Visual Studio 2010 ”，

打开 VS2010。

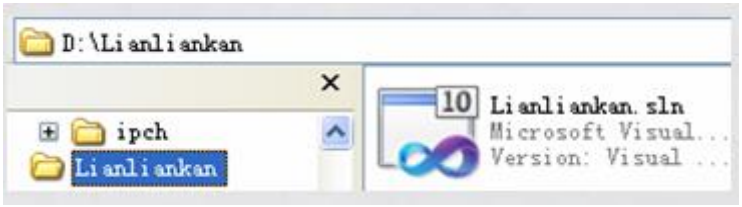
(2) 在 VS2010 开发工具中选“File -> New -> Project”菜单,出现新建对话框。

(3) 在新建对话框中，选择解决方案类型为“Other Project Type -> Visual Studio Soluctions -> Blank Soluction”,解决方案名为“Lianliankan”，保存路径。




**图**错误!文档中没有指定样式的文字。**-4 创建解决方案**

(4) 创建完成后，解决方案保存路径中，生成解决方案文件夹，在解决方案文件夹中，生成解决方案文件(.sln 后缀)。

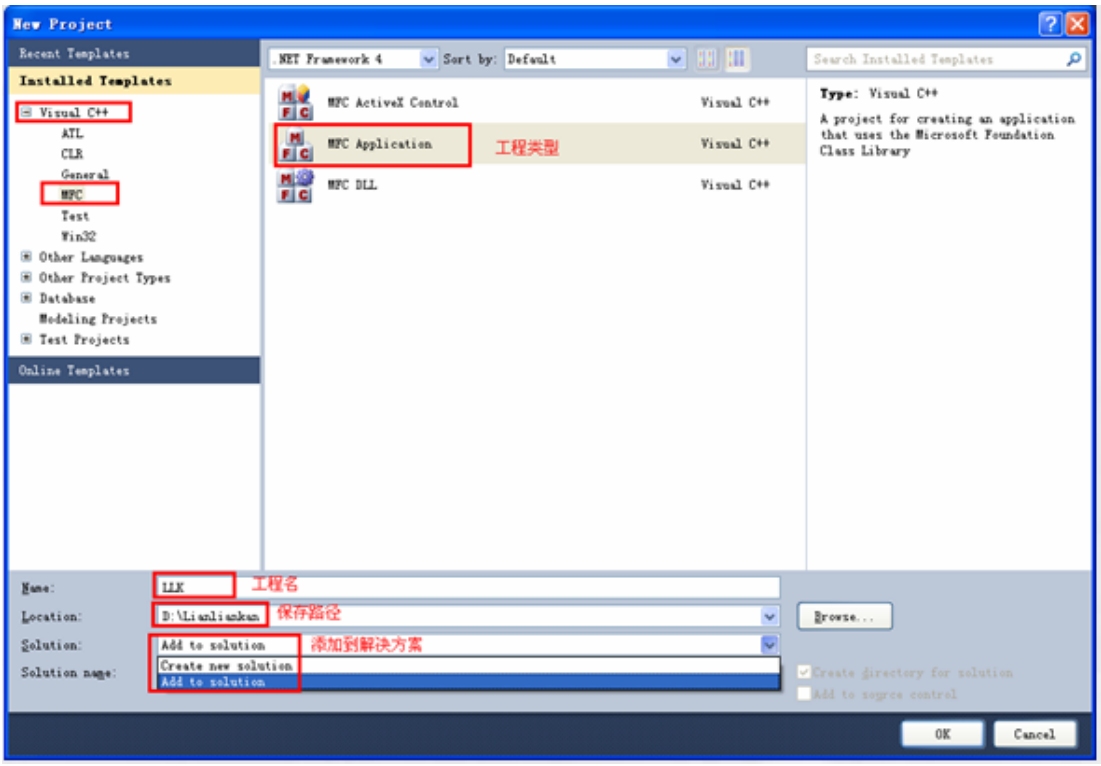



**图**错误!文档中没有指定样式的文字。**-5 生成解决方案的文件**

**2、创建工程**

(1) 创建解决方案之后，选择“File -> New -> Project”，显示新建对话框。

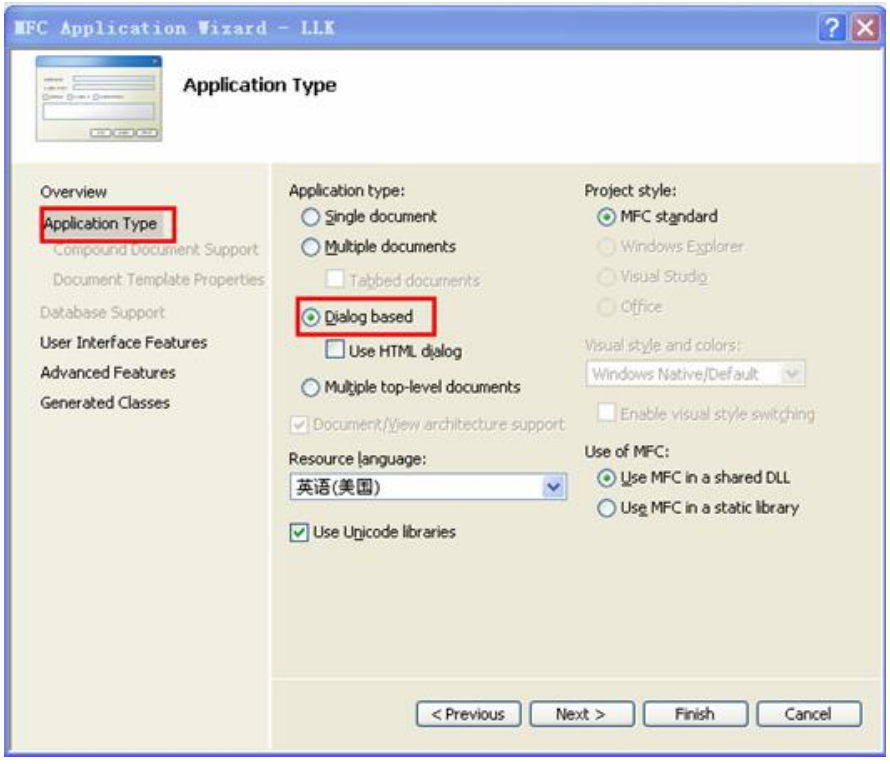
(2) 选择工程类型为“Visual C++ -> MFC -> MFC Application”，输入工程名称 LLK，选择“Soluction”为 “Add to solution”，点击“OK”，进入应用程序向导。




**图**错误!文档中没有指定样式的文字。**-6 创建工程**

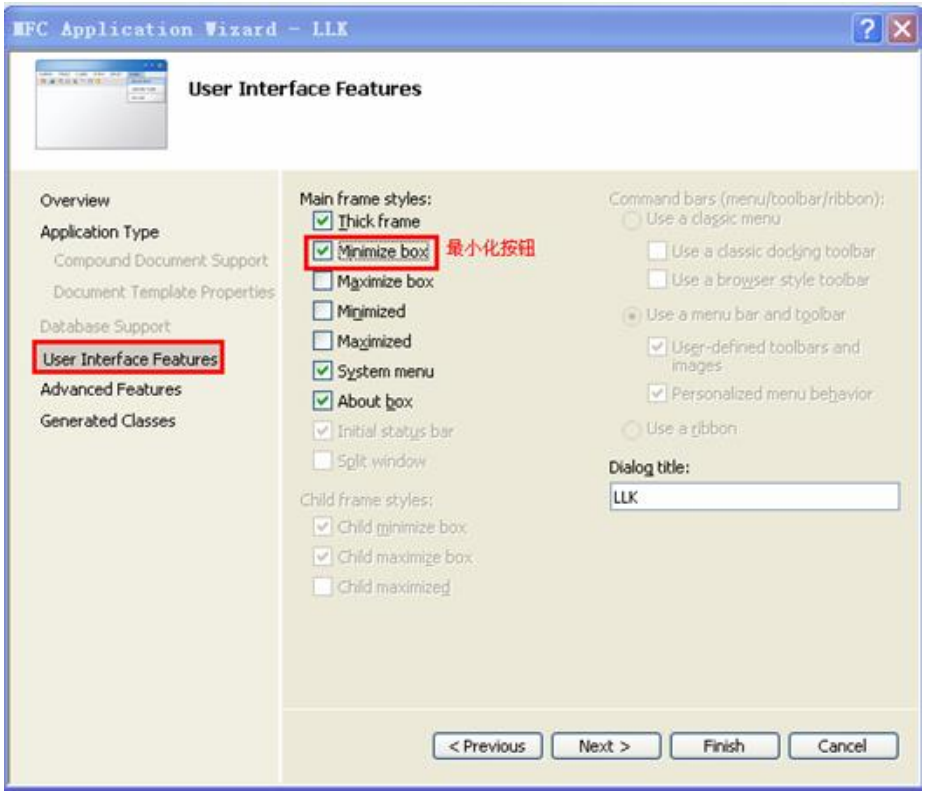
(3) 选择应用程序类型

在应用程序向导的“Application Type”中，选择应用程序类型为“Dialog based”。然后点击“Next”进入下一步。




**图**错误!文档中没有指定样式的文字。**-7 选择应用类型**

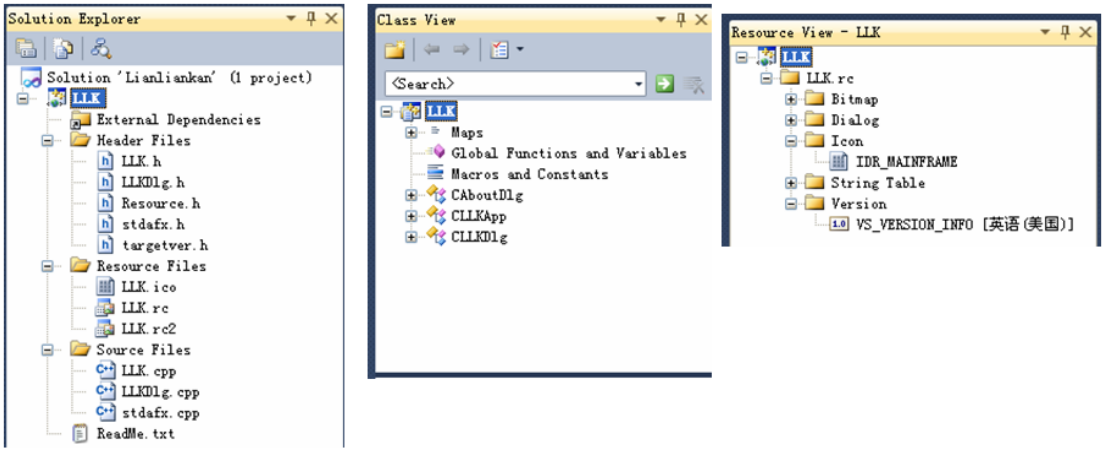
(4) 在“User Interface Features”中，勾选“Minisize box”，给对话框窗口添加一个最小化按钮。然后点击“Finish”完成工程的创建。




**图**错误!文档中没有指定样式的文字。**-8 添加最小化按钮**

(5) 创建完成后，在工程中可以看到如下内容。

**解决方案资源管理器** **类视图** **资源视图**




**图**错误!文档中没有指定样式的文字。**-9 程序的三种视图**

(6) 在工程保存路径下可以看到如下文件：

1) 解决方案文件夹


**图**错误!文档中没有指定样式的文字。**-10 解决方案文件夹**

2) 工程文件夹

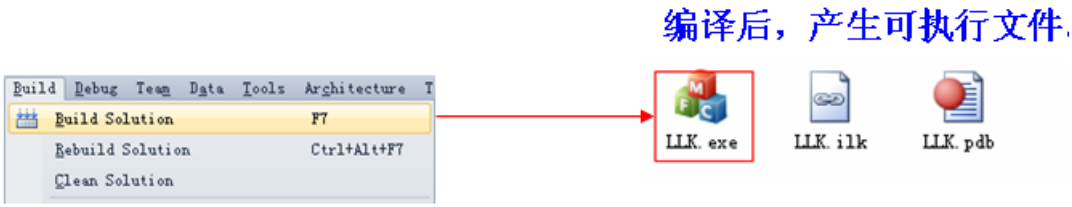

**图**错误!文档中没有指定样式的文字。**-11 工程文件夹**

(7) 编译并运行程序。

使用 VS2010 应用程序向导创建好工程之后，也就完成的代码的编辑工作。现在可以对代码进行编译、连接和运行。

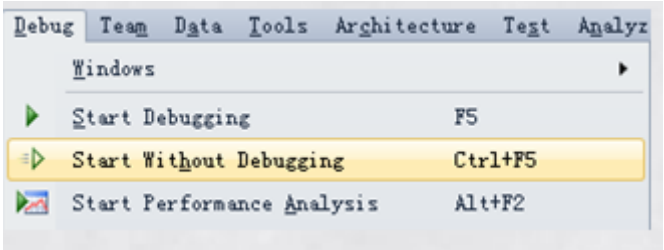
1) 编译程序(F7)

在 VS2010 中，将编译和连接的功能放到一起，直接编译就可以产生可执行文件。




**图**错误!文档中没有指定样式的文字。**-12 产生可执行文件**

2) 运行程序(ctrl+F5)




**图**错误!文档中没有指定样式的文字。**-13 运行程序**

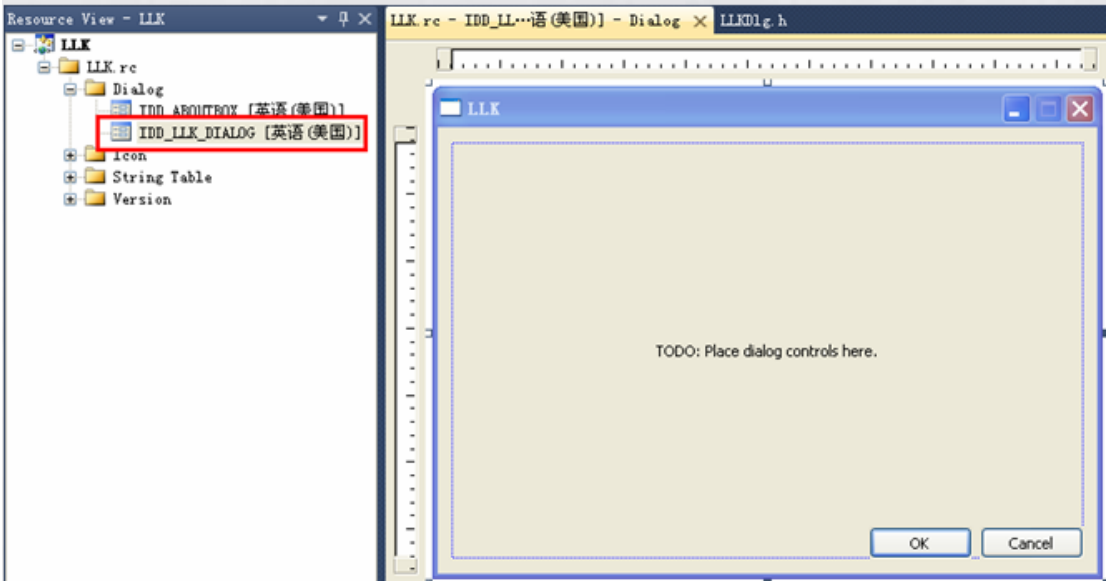
**3、修改主界面对话框属性**

(1) 打开主界面对话框资源

方法一：选择主界面对话框类 CLLKDlg，右键选择“Go To Dialog”,打开主界面对话框资源。


**图**错误!文档中没有指定样式的文字。**-14 打开对话框方法一**

方法二：在资源视图，双击主界面对话栏资源。打开对话框资源。




**图**错误!文档中没有指定样式的文字。**-15 打开对话框方法二**

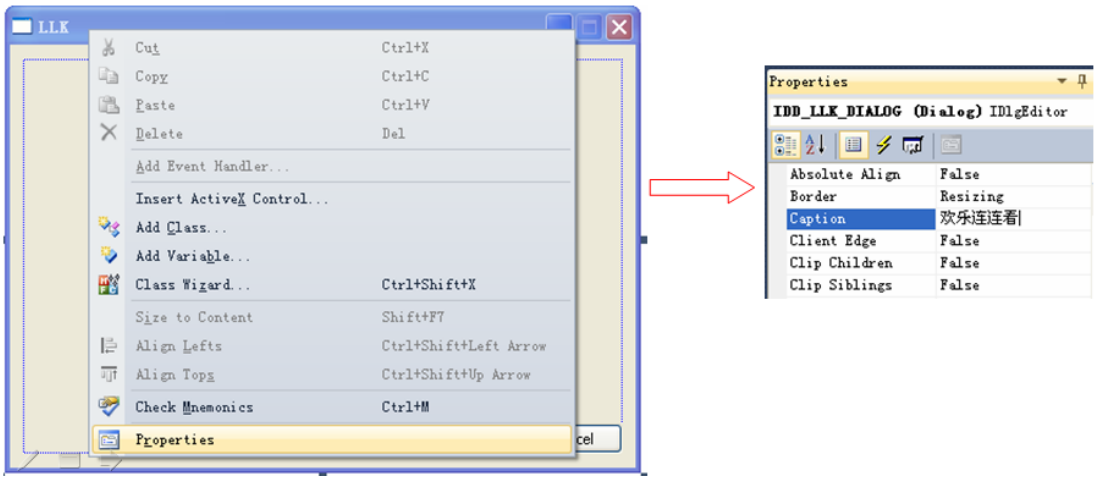
(2) 修改对话框标题为“欢乐连连看”。

1) 在对话对话编辑器中，删除对话框资源中默认产生的控件。

2) 在对话框资源上右键，选择“Properties”，打开对话框属性编辑器。

3) 在对话框属性编辑器中修改对话框标题栏为“欢乐连连看”。

4) 编译并运行程序。




**图**错误!文档中没有指定样式的文字。**-16 修改标题的运行结果**

(3) 修改对话框图标。

1) 在工程目录 res 文件夹中，找到对话框图标“LLK.ico”。将需要设置为对话框图片的 ico 文件命名为“LLK.ico”，替换工程目录 res 中默认的 LLK.ico 文件。


**图**错误!文档中没有指定样式的文字。**-17 修改图标**

2) 编译并运行程序。

由于修改了资源文件，必须要先把原来编译的文件清除后，全部重新编译才行。否则 VS 中默认是增量编译的，已编译的内容不会重新编译。只替换了图标的文件，工具并不会重新编译图标。

# 1.2 主界面

## 1.2.1 功能需求

设计“欢乐连连看”项目的主界面，在主界面上添加一个背景图片，并在适当的地方添加“基本模式”、“休息模式”、“关卡模式”、“帮助”、“排行榜”与“设置”按钮。

1、主界面为启动游戏时出现的界面。


**图**错误!文档中没有指定样式的文字。**-18 主界面效果图**

2、主界面布局

(1) 主界面客户区大小：800*600，单位像素。

(2) 背景图片：背景图片大小为 800*600，背景图片看起来欢快、轻松。


**图**错误!文档中没有指定样式的文字。**-19 位图格式的背景图片**

(3) 模式选择按钮：基本模式、休息模式、关卡模式。

(4) 辅助功能按钮：排行榜、设置、帮助。

(5) 界面要求：界面协调、美观、符合大部分玩家操作习惯。

## 1.2.2 设计思路

**在“创建工程”的基础上进行迭代开发。**

使用 VS2010 工具中对话框编辑器，布置主界面，窗口的背景图一个位图文件，使用 MFC 中的 GDI 技术来给窗口添加一个背景。

**1、背景图片绘制步骤**

(1) 背景图片的导入

在工程的资源视图中，导入背景图片，背景图片资源 ID 为 IDB_MAIN_BG。

(2) 背景图片的显示

1) 在 CLLKDlg 类中，添加一个初始化背景的方法 void InitBackground()函数，将位图选入位图内存。

2) 在 CLLKDlg::OnInitDialog()函数中，调用 InitBackground()函数。在对话框启动时，将位图选进位图内存。

3) 在主界面窗口的 WM_PAINT 消息响应函数 CLLKDlg::OnPaint()函数中，将选入位图内存中的图片选到视频内存中，然后调用 CDC::BitBlt()函数，将位图显示在主界面上。

**2、CLLKDlg 类设计**

(1) 属性

CDC m_dcMem; 表示内存 DC，访问权限为 protected。

(2) 方法

1) void InitBackground()

将背景图片位图资源保存到位图内存中。

2) void Paint()

对话框 WM_PAINT 消息响应函数，将位图内存中的背景图片绘制到对话框中。

3) BOOL OnInitDialog()

对话框初始化函数，在对话框初始化时，调用 InitBackground()函数，加载位图资源到位图资源中。

## 1.2.3 编码实现

**导入“创建工程”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：位图导入。

步骤二：绘制窗口背景。

步骤三：添加主界面的功能按钮。

**1、位图导入**

(1) 将位图资源文件放到物理磁盘工程目录下的 res 文件夹。

(2) 将位图资源导入到工程中。

(3) 修改位图资源为 IDB_MAIN_BG。


**图**错误!文档中没有指定样式的文字。**-20 位图导入**

**2、绘制窗口背景**

(1) 创建一个内存 DC

给 CLLKDlg 类添加一个 protected 属性 CDC m_dcMem，表示位图内存。

(2) 在 CLLKDlg 类添加 void InitBackground()函数。

```
void CLLKDlg::InitBackground(void)
{
	//……
}
```

(3) 加载位图

CBitmap 类封装了 Windows 的图形设备(GDI)中的位图，并且提供了操纵位图的成员函数。静态位图的加载：

1) 定义 CBitmap 位图对象。
2) 调用 CBitmap::LoadBitmap()函数加载位图资源。
3) 将位图选入到内存 DC 中。

加载背景图片位图代码如下：

```
CBitmap bmpMain;
bmpMain.LoadBitmapW(IDB_MAIN_BG);
```

(4) 创建兼容 DC

在对话框的非 OnPaint()函数中，可以使用 CClientDC 类来得到当前对话框的视频内存 DC。通过调用 CDC::CreateCompatibleDC()函数来创建一个与视频 DC 兼容的内存 DC（m_dcMem），最后，将位图对象选入（CDC::SelectObject）到内存 DC 中。

```
void CLLKDlg::InitBackground(void)
{
	// 获得当前对话框的视频内存
	CClientDC dc(this);
	//……
	// 创建与视频内存兼容的内存 DC
	m_dcMem.CreateCompatibleDC(&dc);
	// 将位图资源选入 DC
	m_dcMem.SelectObject(bmpMain);
```

```
    //……
}
```

(5) 在 CLLKDlg::OnInitDialog()函数中，调用 InitBackground()函数。

(6) 调用 CDC::BitBlt()函数。

在 CLLKDlg::OnPaint()函数中，调用 CDC::BitBlt()函数从位图内存中拷贝到视频内存中，进行显示。

```
void CLLKDlg::OnPaint()
{
    if (IsIconic())
    {
        //……
    }
    else
    {
        // 创建 CPaintDC 对象
        CPaintDC dc(this);
        // 绘制背景图片
        dc.BitBlt(0, 0, 800, 600, &m_dcMem, 0, 0, SRCCOPY);

        CDialogEx::OnPaint();
    }
}
```

**3、添加主界面的功能按钮**

(1) 给界面添加控件

在对话框编辑器中，将背景图片导入，调整对话框的大小。从 Toolbox 中将按钮控件添加到对话框中合适的位置。


VS工具的Toolbox中，有界面布局所需要用到的所有控件。用鼠标选中控件并拖到对话框中，即可将控件添加到对话框中。

**图**错误!文档中没有指定样式的文字。**-21 添加功能按钮**

(2) 修改按钮文本(Caption)和 ID。

鼠标右键选择控件，选择“Properties”，可以打开属性编辑器。在属性编辑器中可以查看和修改控件属性。

控件 ID 由大小字母和下划线组成，切忌用中文。

| ID | Caption |
|---|---|
| IDC_BTN_BASIC | 基本模式 |
| IDC_BTN_RELAX | 休息模式 |
| IDC_BTN_LEVEL | 关卡模式 |
| IDC_BTN_RANK | 排行榜 |
| IDC_BTN_SETTING | 设置 |
| IDC_BTN_HELP | 帮助 |

(3) 编译运行，调整对话框大小和控制位置。

(4) 设置主界面客户区的大小。

通过调用 MoveWindow()函数可以设置窗口的位置和大小。在 InitBackground 函数中添加如下代码，以调整窗口的大小。

```
void CLLKDlg::InitBackground(void)
{
	//……
	// 调整窗口大小
	CRect rtWin;
	CRect rtClient;
	this->GetWindowRect(rtWin);		// 获得窗口大小
	this->GetClientRect(rtClient);		// 获得客户区大小
	// 标题栏和外边框的大小
	int nSpanWidth = rtWin.Width() - rtClient.Width();
	int nSpanHeight = rtWin.Height() - rtClient.Height();

	// 设置窗口大小
	MoveWindow(0, 0, 800 + nSpanWidth, 600 + nSpanHeight);
	//……
}
```

(5) 调用 CenterWindow()函数，使窗口居中。

## 1.3 开始游戏

### 1.3.1 功能需求

当玩家在主界面选择【基本模式】时，出现基本游戏界面，并隐藏主界面。

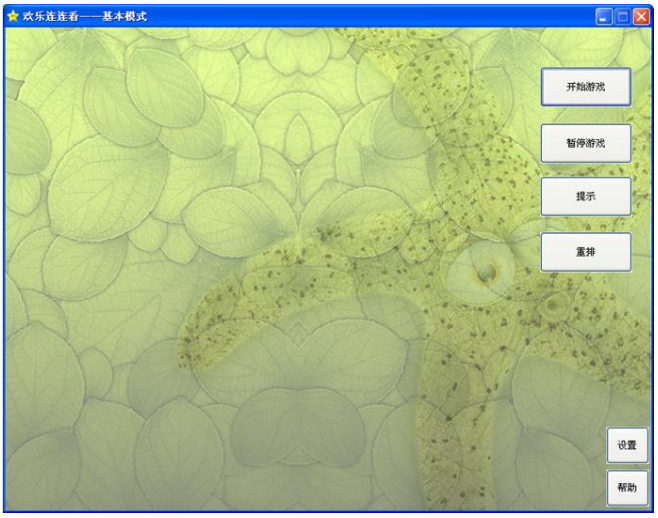
玩家点击【开始游戏】按钮，生成游戏地图。基本模式的游戏地图为 10 行 16 列，图片的花色 16 个。

**1、客户区**

(1) 背景图片。

(2) 开始游戏按钮、暂停按钮、提示按钮、重排按钮、设置按钮、和帮助按钮。

(3) 当选择最小化按钮时，将游戏界面最小化到窗口左下角。




**图**错误!文档中没有指定样式的文字。**-22 界面效果**

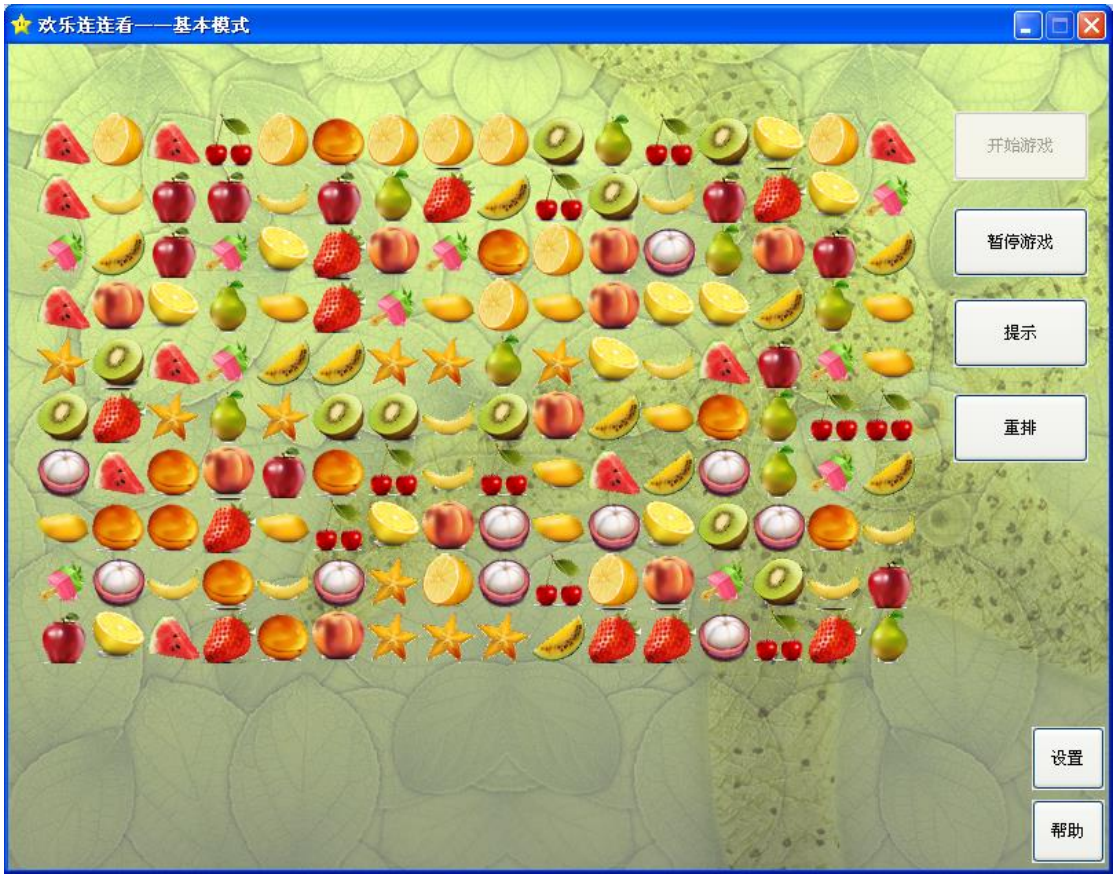
**2、游戏地图区域**

(1) 游戏地图起始点 (50,50)，单位像素。

(2) 游戏地图：10 行，16 列。

(3) 每张图片大小：40*40，单位像素。

(4) 游戏地图中包含 16 种图片。




**图**错误!文档中没有指定样式的文字。**-23 游戏地图**

**3、游戏元素**

(1) 游戏元素如下图所示(图片根据确定的主题自定义)。

1) 图片背景为白色。
2) 单张图片为 40*40，单位像素。
3) 元素图片由单个小图片按顺序拼接得到。

(2) 例如前 6 个图片元素，分别用 0、1、2、3、4、5 来表示这 6 种图片。


**图**错误!文档中没有指定样式的文字。**-24 游戏元素**

(3) 物理磁盘保存路径

图片名：fruit_element.bmp，放在 LLK 工程目录下 theme\picture 文件夹中。图片为一张 40*800 的 BMP 图片，该图片中由 20 种大小为 40*40 的小图片组成。

**图**错误!文档中没有指定样式的文字。**-25 游戏元素的保存**

## 1.3.2 设计思路

**在“主界面”的基础上进行迭代开发。**

**1、数据设计**

添加 global.h 文件，定义结构体 Vertex，用于保存游戏地图中一个点的行号、列表、花色。

**2、去除元素背景**

(1) 图片在内存中以像素点存储，用 RGB 的方式表示。

RGB 的各项的值范围为：0~255，使用 8 位二进制的 bit 表示，即一个字节。因此，一个像素用 3 个字节来存储。

**如，某一像素点颜色的RGB值为(224, 162, 67)**
**则，在内存中的值为：110000010100010001000011**


| RGB | R | | | | | | | | G | | | | | | | | B | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 十进制 | 224 | | | | | | | | 162 | | | | | | | | 67 | | | | | | | |
| 十六进制 | E | | | | 0 | | | | A | | | | 2 | | | | 4 | | | | 3 | | | |
| 内存中的值 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

**RGB换算**

**图**错误!文档中没有指定样式的文字。**-26 图片存储和表示**

(2) 黑色的 RGB 值为 RGB(0, 0, 0)，白色的 RGB 值为 RGB(255, 255, 255)。

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 黑色 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 白色 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

可以看出，黑色为全0，白色为全1，其它的颜色为0或1的组合
所以图片的数据在内存中实际上就是0与1的组合

**图**错误!文档中没有指定样式的文字。**-27 黑和白 RGB 值**

(3) BitBlt()函数对图片的操作，实际上是对像素进行位块的操作。

常用位操作包括：与( & )、或( | )。

BitBlt()函数位操作类型由最后一个参数，光栅码来控制。参数意思为：

与(SRCAND)：D1 = D1 & D2。

或(SRCPAINT)：D1 = D1 | D2。

拷贝(SRCCOPY)：D1 = D2。

| A | B | P = A & B |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |

| A | B | P = A \| B |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 1 |

**图**错误!文档中没有指定样式的文字。**-28 位操作**

(4) 利用掩码消除图片背景的原理

| | 显示DC<br>m_dcMem<br>D1 | 掩码DC<br>m_dcMask<br>D2 | 显示DC<br>m_dcMem<br>D1 = D1 \| D2 | 元素DC<br>n_dcElement<br>D3 | 显示DC<br>m_dcMem<br>D1 = D1 & D3 |
|---|---|---|---|---|---|
| 图片的白边区域 | X0 | 0 | X0 | 1 | X0 |
| 图片元素区域 | X0 | 1 | 1 | X | X |


**图**错误!文档中没有指定样式的文字。**-29 消除图片背景**

### 3、类设计

(1) CLLKDlg 类

CLLKDlg 类为主界面对话框类。在该类中添加一个“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。

(2) CGameDlg 类

显示游戏界面，并与用户交换信息，不关心数据是以何种结构存储到内存中。CGameDlg

类为游戏界面对话框类。

1) 数据成员

| 成员变量 | 描述 |
| --- | --- |
| CDC m_dcMem | 内存 DC |
| CDC m_dcBG | 背景 DC |
| CDC m_dcElement | 元素内存 DC |
| CDC m_dcMask | 掩码内存 DC |
| CPoint m_ptGameTop | 游戏区起始点（游戏第一张图片的顶点坐标,坐标相对于窗口客户区） |
| CSize m_sizeElem | 元素图片的大小 |
| CRect m_rtGameRect | 游戏区域大小(该区域为游戏更新区域大小，考虑到后面画的线，可能会超出图片区域一定范围) |
| CGameControl m_GameC | 游戏控制类 |

2) 成员函数

| 成员函数 | 描述 |
| --- | --- |
| virtual BOOL OnInitDialog() | 窗口初始化函数 |
| afx_msg void OnPaint() | WM_PAINT 消息函数 |
| afx_msg void OnBnClickedBtnStart() | 开始按钮 |
| void InitBackground(void) | 初始化窗口背景 |
| void InitElement(void) | 初始化元素图片与 DC |
| void UpdateWindow(void) | 更新界面 |
| void UpdateMap(void); | 更新游戏地图 |

(3) CGameControl 类

保存游戏数据，统一协调和调用各逻辑类完成相应的功能。

1) 静态成员变量

| 静态成员变量 | 描述 |
| --- | --- |
| static int s_nRows | 游戏行数 |
| static int s_nCols | 游戏列数 |
| static int s_nPicNum | 图片数 |

2) 数据成员

| 数据成员 | 描述 |
| --- | --- |
| int** m_pGameMap | 游戏地图数组指针 |
| CGameLogic m_GameLogic | 游戏逻辑操作对象 |

3) 成员函数

| 成员函数 | 描述 |
| --- | --- |
| void StartGame(void) | 开始游戏 |
| int GetElement(int nRow, int nCol) | 得到某一个元素 |

(4) CGameLogic 类

根据不同的数据结构，完成数据逻辑的判断功能。

| 成员函数 | 描述 |
|---|---|
| int** InitMap() | 初始化游戏地图 |
| void ReleaseMap(int** &pGameMap) | 释放地图 |

(5) CGameException 类

游戏异常类，处理程序中的一些异常。程序中类是层层调用的，当下一层出现异常，则可以抛出一个异常对象。在上一层，则使用 try{}catch{}语句来捕获异常。

## 1.3.3 编码实现

**导入“主界面”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：添加游戏对话框资源。

步骤二：创建并显示游戏对话框。

步骤三：绘制游戏界面背景。

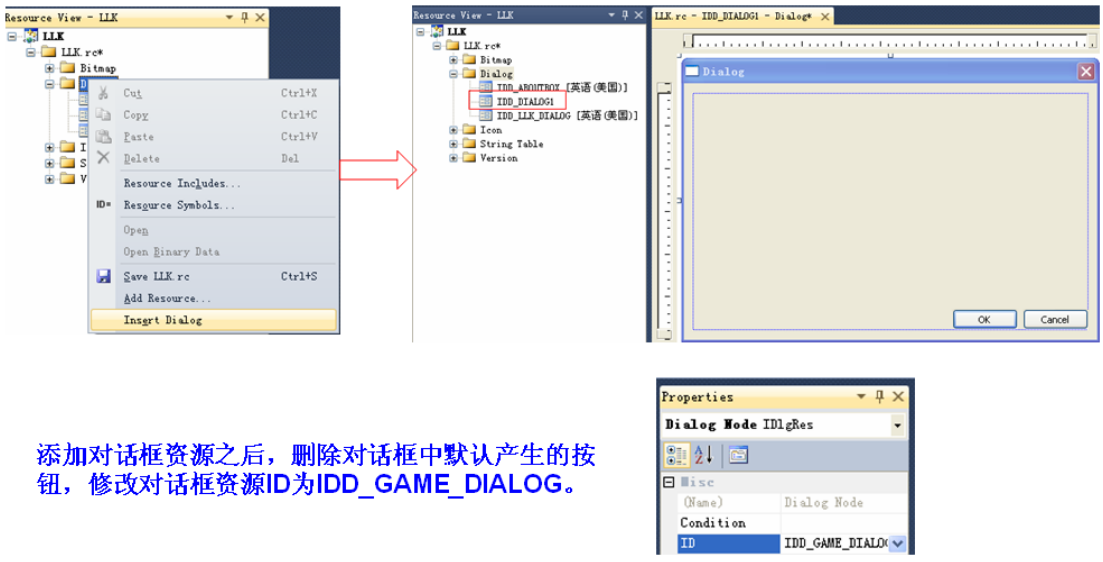
步骤四：游戏界面布局。

步骤五：加载游戏元素图片。

步骤六：绘制游戏地图。

**1、添加游戏对话框资源**

在资源视图中，右键选择“Dialog”，在弹出菜单中选择“Insert Dialog”菜单项。即可添加一个对话框资源。




**图**错误!文档中没有指定样式的文字。**-30 添加游戏对话框资源**

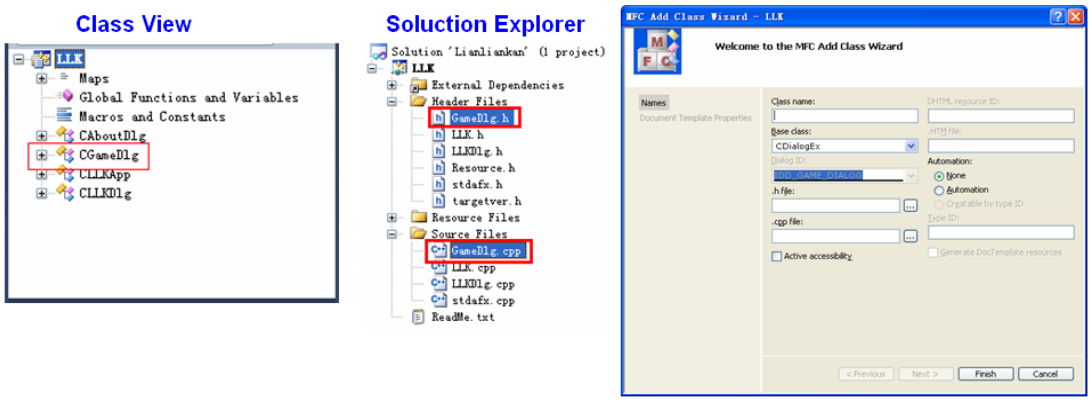
**2、创建并显示游戏对话框**

(1) 添加游戏界面对话框类 CGameDlg。

1) 双击对话框资源，弹出“MFC Add Class Wizard”对话框。

2) 在对话框中输入对话框类的类名为“CGameDlg”。点击“Finish”，完成添加。

3) 添加完成后，在“Class View”可以看到添加 CGameDlg 类，在“Soluction Explorer”可以看到 CGameDlg 类的源文件和头文件。

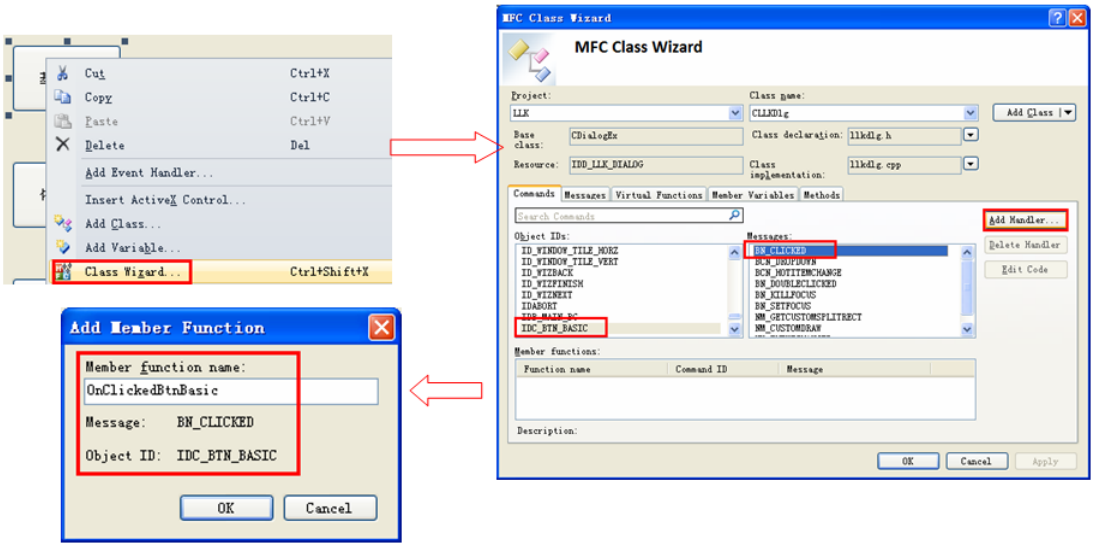



**图**错误!文档中没有指定样式的文字。**-31 创建游戏对话框类**

在 MFC 中，对话框资源和对话框类是通过对话框的资源 ID 进行关联的，每个对话框类都有一个对应的对话框资源。

(2) 创建并显示游戏对话框

1) 使用“Class Wizard”，给主界面对话框中的“基本模式”按钮(IDC_BTN_BASIC)，添加 BN_CLICKED 消息响应函数 OnBnClickedBtnBasic()。




**图**错误!文档中没有指定样式的文字。**-32 运行程序**

2) 在 CLLKDlg::OnBnClickedBtnBasic()函数中，调用 CGameDlg::DoModel()函数创建并显示基本游戏对话框。

```
void CLLKDlg::OnBnClickedBtnBasic()
{
    CGameDlg dlg;
    dlg.DoModal();
```

```
}
```

3) 控制主界面的隐藏与显示

在 CLLKDlg::OnBnClickedBtnBasic()函数中，调用 CWnd::ShowWindow()函数，来控制主界面的隐藏和显示。

**3、绘制游戏界面背景**

(1) 加载游戏界面背景图片

1) 位图的加载方式

方式一：静态方式，位图直接导入到工程中。

如在“主界面设计”中，由于主界面的背景不会改变，这时，就可以使用这一种方式，直接将要显示的位图导入到工程中，在工程中，通过资源 ID 加载位图。

优点：可直接使用 CBitmap 类的 LoadBitmap()函数完成。缺点：图片是不可修改的。

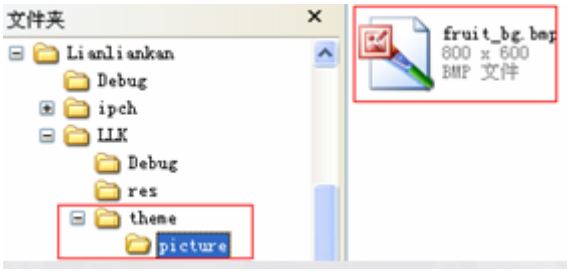
方式二：动态方式，通过传入的位图文件名加载位图。

由于“欢乐连连看”游戏，是可以更换主题的，即背景与元素图片都是可以在程序运行时修改的，这样，使用静态的方式就会很不灵活，也无法实现用户自定义。

优点：使用程序很灵活，可以通过修改位图文件名，改变加载的位图。缺点：需要用到系统 API 函数 LoadImageW()函数来实现，程序会复杂一点。

2) 采用动态方式加载位图

在工程目录下新建一个“theme”文件夹，将游戏背景图片放到该文件夹中的“picture”文件夹中。




**图**错误!文档中没有指定样式的文字。**-33 放置背景图片**

给 CGameDlg 类添加成员函数 void InitBackground()，在该函数中调用 Windows API 函数 LoadImage()加载位图。

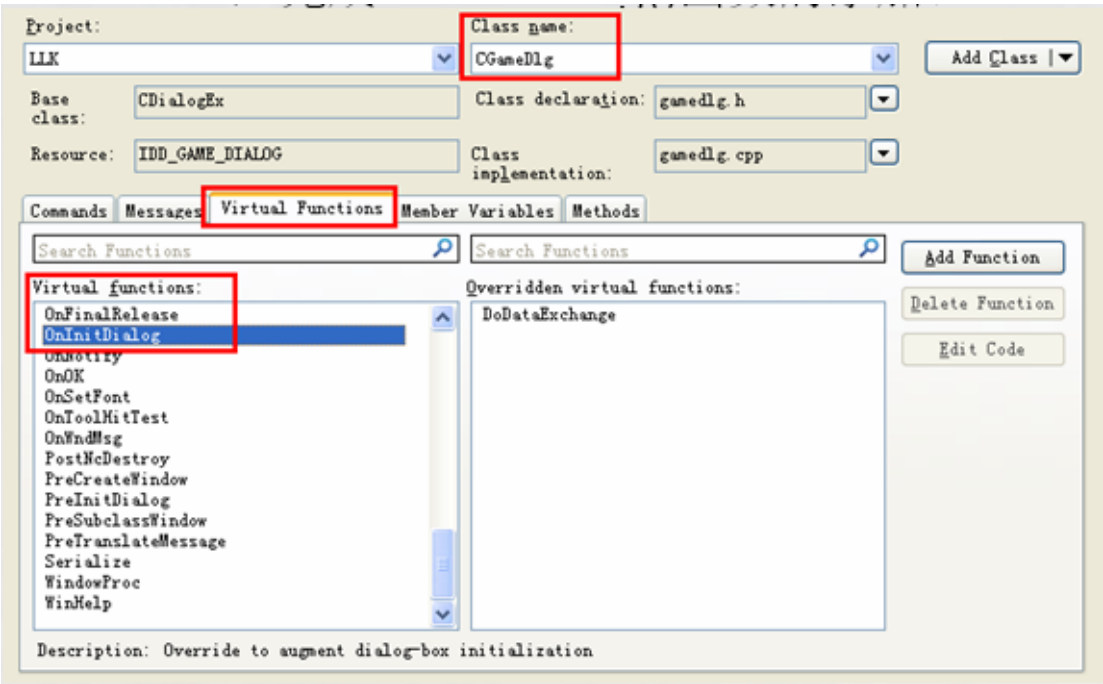
(2) 将图片选入位图内存。

在对话框初始化时，将位图保存到位图内存中。给 CGameDlg 类添加 OnInitDialog()函数，在 OnInitDialog()函数中，调用 InitBackground()函数。

1) 右键选择“CGameDlg”类，在弹出菜单中选择“Class Wizard”。

2) 在“Class Wizard”对话框中，选择“Vritual Functions”标签页，选择添加的函数为“OnInitDialog”。

3) 点击“Add Function”，完成 OnInitDialog()函数的添加。




**图**错误!文档中没有指定样式的文字。**-34 添加 OnInitDialog 函数**

(3) 将图片从位图内存拷贝到视频内存。

给 CGameDlg 类添加 OnPaint()函数，在绘制界面时，从位图内存中拷贝位图到视频内存，进行显示。

1) 打开 CGameDlg 类的 Class Wizard。

2) 在 Class Wizard 中选择“Messages”标签页，选择“WM_PAINT”消息。点击“Add Handler”按钮，完成 WM_PAINT 消息响应函数 OnPaint()的添加。

3) 在 CGameDlg::OnPaint()函数中，调用 CPaintDC::BitBlt()函数，绘制游戏界面背景。

```
void CGameDlg::OnPaint()
{
	CPaintDC dc(this); // device context for painting

	// 绘制背景图片
	dc.BitBlt(0, 0, GAMEWND_WIDTH, GAMEWND_HEIGHT, &m_dcMem, 0, 0, SRCCOPY);
}
```

(4) 添加 CGameDlg::UpdateWindow()函数，调整游戏窗口大小。

设置游戏界面客户区大小为 800*600，在 OnInitDialog()中调用 UpdateWindow()函数。

```
void CGameDlg::UpdateWindow(void)
{
	// 调整窗口大小
	CRect rtWin;
	CRect rtClient;
	this->GetWindowRect(rtWin);		// 获得窗口大小
```

```
    this->GetClientRect(rtClient);          // 获得客户区大小

    // 标题栏和外边框的大小
    int nSpanWidth = rtWin.Width() - rtClient.Width();
    int nSpanHeight = rtWin.Height() - rtClient.Height();

    // 设置窗口大小
    MoveWindow(0, 0, GAMEWND_WIDTH + nSpanWidth, GAMEWND_HEIGHT + nSpanHeight);

    // 设置对话框显示是，在 windows 窗口正中央。
    CenterWindow();
}
```

**4、游戏界面布局**

(1) 设置游戏界面对话框标题

在 CGameDlg::OnInitDialog()函数中，设置对话框的标题为“欢乐连连看——基本模式”。

(2) 设置游戏界面对话框图标

(3) 添加控件

在对话框中添加控件，并调整按钮在对话框中的位置。

**5、加载游戏元素图片**

(1) 将游戏元素图片加载到程序中

1) 给 CGameDlg 类添加成员函数 CGameDlg::InitElement(void)，访问权限为 protected。

2) 调用::LoadImageW()函数，将游戏元素图片加载到程序中。

3) 创建元素图片的位图内存 DC。

CGameDlg 类添加 CDC 数据成员 m_dcElement，m_dcElement 与视频 DC 兼容。

4) 将位图资源选入创建的 DC。

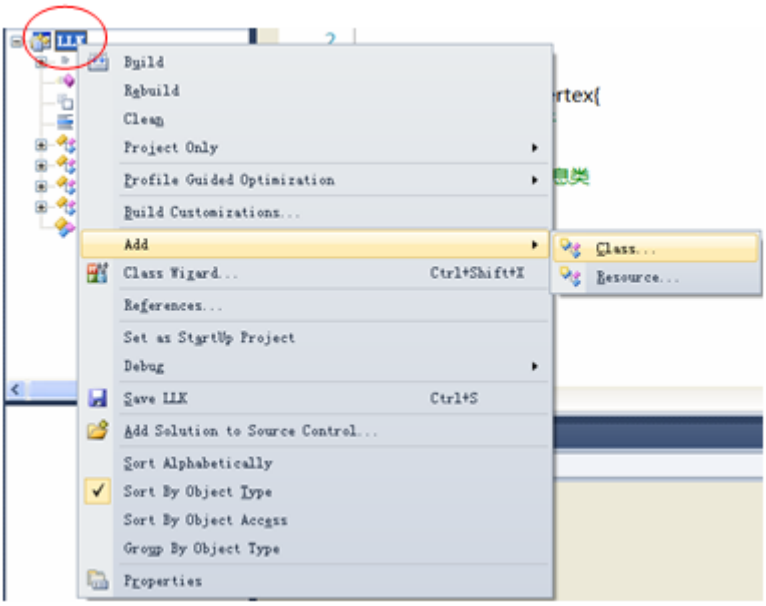
5) 在对话框初始化时，调用 InitElement()函数。

```
void CGameDlg::InitElement(void)
{
    // 获得当前对话框的视频内存
    CClientDC dc(this);

    // 加载 BMP 图片资源
    HANDLE hBmp = ::LoadImageW(NULL, _T("theme\\picture\\fruit_element.bmp"),
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);

    // 创建与视频内存兼容的内存 DC
```

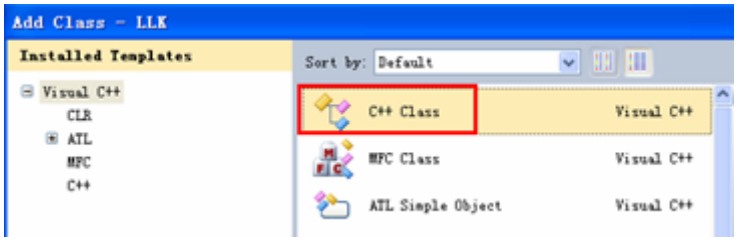
```
        m_dcElement.CreateCompatibleDC(&dc);
        // 将位图资源选入 DC
        m_dcElement.SelectObject(hBmp);

        // 加载掩码 BMP 图片资源
        HANDLE hMask = ::LoadImageW(NULL, _T("theme\\picture\\fruit_mask.bmp"),
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);

        // 创建与视频内存兼容的内存 DC
        m_dcMask.CreateCompatibleDC(&dc);
        // 将位图资源选入 DC
        m_dcMask.SelectObject(hMask);
}
```

(2) 添加 CGameLogic 类

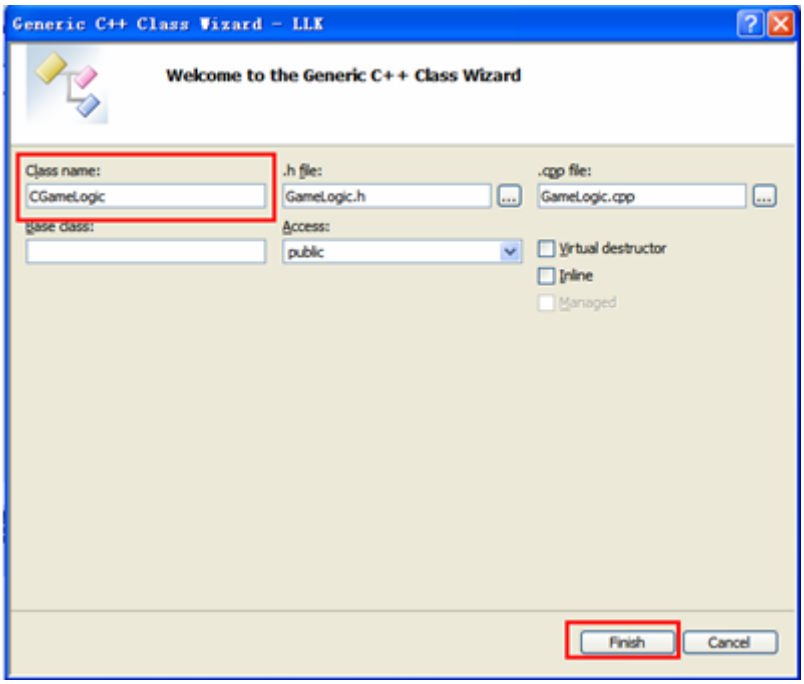
1) 切换到类视图，在工程上右击，在弹出菜单中选择“Add” -> “Class”。




2) 选择类的类型：C++ Class，点击“Add”。




3) 在 Class name 中输入类名：CGameLogic，点击“Finish”，完成类的创建。




(3) 在 CGameLogic 类中添加初始化游戏地图函数

1) 获取地图花色和大小

```
int** CGameLogic::InitMap()
{
	// 获取地图大小和花色
	int nRows = CGameControl::s_nRows;
	int nCols = CGameControl::s_nCols;
	int nPicNum = CGameControl::s_nPicNum;
}
```

2) 为游戏地图开辟内存空间

```
int** CGameLogic::InitMap()
{
	//……
	// 开辟内存区域
	int** pGameMap = new int*[nRows];
	if(NULL == pGameMap)
	{
		throw new CGameException(_T("内存操作异常！ \n"));
	}
	else
	{
		for (int i = 0; i < nRows; i++)
		{
			pGameMap[i] = new int[nCols];
			if(NULL == pGameMap)
```

```
            {
                throw new CGameException(_T("内存操作异常！\n"));
            }
            memset(pGameMap[i], NULL, sizeof(int) * nCols);
        }
    }
}
```

3) 根据花色的种类计算出每种花色的图片的平均个数，依次给数组赋值。

```
int** CGameLogic::InitMap()
{
    //……
    // 多少花色
    if ((nRows * nCols) % (nPicNum * 2) != 0)
    {
        ReleaseMap(pGameMap);
        throw new CGameException(_T("游戏花色与游戏地图大小不匹配！"));
    }
    int nRepeatNum = nRows * nCols / nPicNum;
    int nCount = 0;
    for(int i = 0; i < nPicNum; i++)
    {
        // 重复数
        for(int j = 0; j < nRepeatNum; j++)
        {
            pGameMap[nCount / nCols][nCount % nCols] = i;
            nCount++;
        }
    }
}
```

4) 随机找到两个位置的图片，进行交换。

```
int** CGameLogic::InitMap()
{
    //……
    // 设置种子
    srand((int)time(NULL));

    // 随机任意交换两个数字
    int nVertexNum = nRows * nCols;
    for(int i = 0; i < nVertexNum; i++)
    {
        // 随机得到两个坐标
        int nIndex1 = rand() % nVertexNum;
```

```
            int nIndex2 = rand() % nVertexNum;

            // 交换两个数值
            int nTmp = pGameMap[nIndex1 / nCols][nIndex1 % nCols];
            pGameMap[nIndex1 / nCols][nIndex1 % nCols] = pGameMap[nIndex2 / nCols][nIndex2 % nCols];
            pGameMap[nIndex2 / nCols][nIndex2 % nCols] = nTmp;
        }

        return pGameMap;
}
```

(4) 在 CGameLogic 类中创建释放游戏地图函数：void ReleaseMap(int** &pGameMap);

```
void CGameLogic::ReleaseMap(int** &pGameMap)
{
    for (int i = 0; i < CGameControl::s_nRows; i++)
    {
        delete []pGameMap[i];
    }
    delete []pGameMap;
}
```

(5) 调用初始化游戏地图函数，并进行异常处理

1) 添加 CGameControl 类。

2) 在 CGameControl 类中定义游戏地图数组指针：int** m_pGameMap。

3) 在CGameControl类中添加StartGame()函数，调用CGameLogic中的InitMap()函数，初始游戏数组。

(6) 生成地图数据

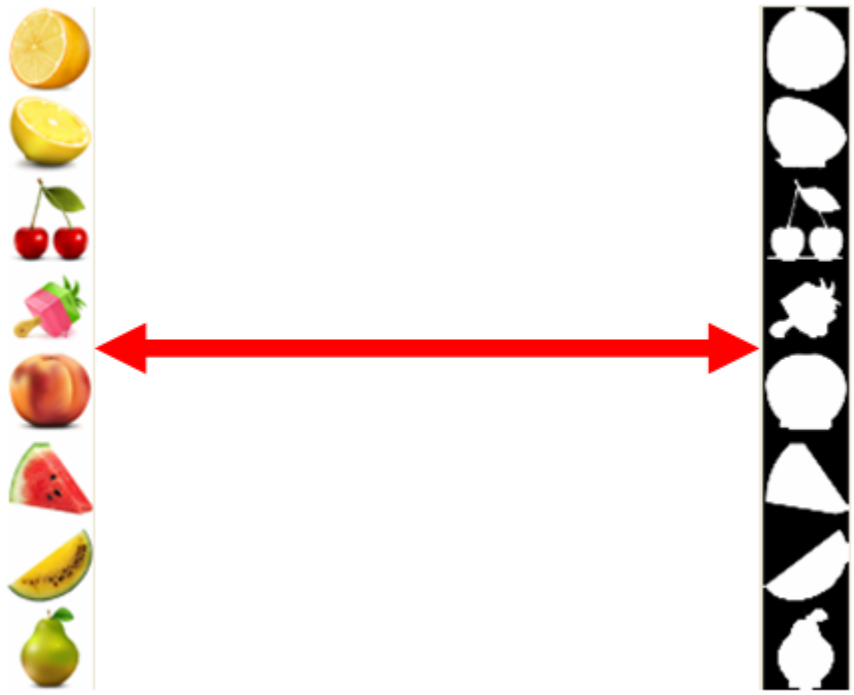
1) 给“开始游戏”按钮，添加 BN_CLICKED 消息响应函数 OnBnClickedBtnStart()。

2) 在 CGameDlg::OnBnClickedBtnStart()中，调用 CGameControl 类中的 StartGame()初始游戏地图的数据。

**6、绘制游戏地图**

(1) 调用 CGameControl 类中的 GetElement()获取相应行列位置图片的元素编号值，并将对应编号的图片区域的数据绘制到 m_dcMem 中的相应位置。

1) 制作一张与元素图片大小一致的掩码图片。掩码图片中，原图中白色的背景改为黑色，原图片区域填成白色。



**图**错误!文档中没有指定样式的文字。**-35 掩码图片**

2) 创建元素图片和掩码图片的位图内存 DC。

3) 先将 m_dcMem 与 m_dcMask 相或(光栅码为：SRCPAINT)，再将此时的 m_dcMem 与元素 m_dcElement 相与(光栅码为：SRCAND)。

(2) 游戏地图的起始点为客户区中的(20,50)。游戏地图分为 10 行 16 列，由 CGameControl 类的静态成员变量 s_nRows 和 s_nCols 得到。每格的大小和元素图片一致，每个元素大小一致(40*40)。

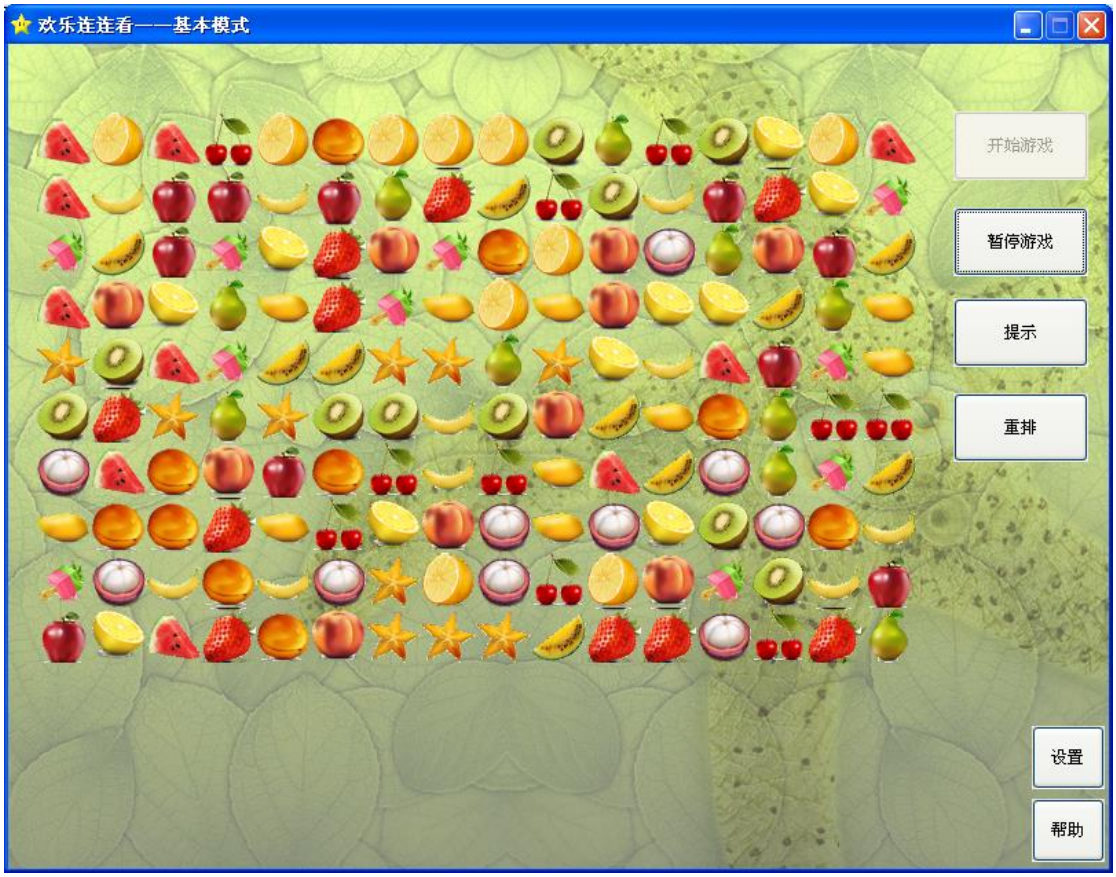
(3) 在 CGameDlg 类中定义 UpdateMap()函数，来绘制游戏界面。

```
void CGameDlg::UpdateMap(void)
{
    // 计算图片的顶点坐标与图片大小
    //……

    //获取行数和列数
    //……

    for(int i = 0; i    < nRows; i++)
    {
        for(int j = 0; j < nCols; j++)
        {
            // 得到图片编号的值
            int nElemVal = m_GameC.GetElement(i, j);
```

```
                // 将背景与掩码相或，边保留，图像区域为 1
                m_dcMem.BitBlt(nLeft + j * nElemW, nTop + i * nElemH , nElemW, nElemH,
                    &m_dcMask, 0, nElemVal * nElemH, SRCPAINT);

                // 与元素图片相与，边保留，图像区域为元素图片
                m_dcMem.BitBlt(nLeft + j * nElemW, nTop + i * nElemH , nElemW, nElemH,
                    &m_dcElement, 0, nElemVal * nElemH, SRCAND);
            }
        }
}
```

(4) 在绘制游戏地图之后，调用 InvalidateRect()函数，更新游戏区域。

(5) 编译运行程序。




**图**错误!文档中没有指定样式的文字。**-36 运行结果**