黑白棋制作—— LabVIEW 界面设计探索

NI 软件工程师　阮奇桢

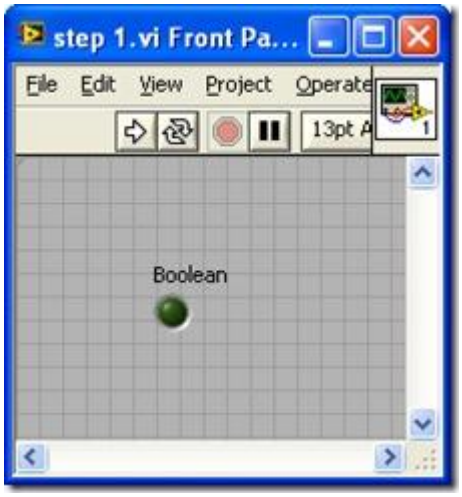
我们需要一个具体示例来帮助介绍这些的技巧，我打算以编写一个黑白棋游戏的界面为例。选择黑白棋是因为这个游戏的界面在常见棋类中比较简单，适合做范例。另外，它也是我最开始学习 LabVIEW 时的练习程序之一，比较有感情:) 黑白棋的棋盘由 8×8 个正方格组成，旗子为黑白两色，放置在方格中。

编写这样一个界面可以使用到多种不同的思路和技巧，我会按照从简到繁的顺序，分几次来介绍几个不同的方法。

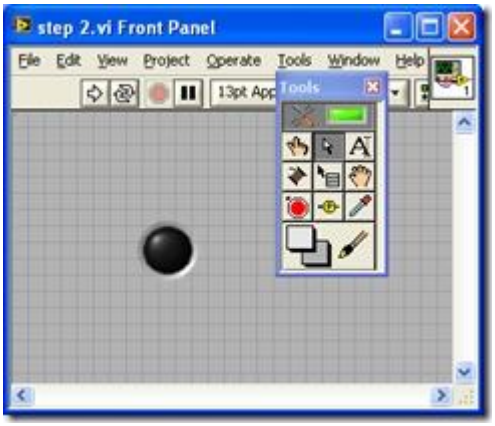
界面设计的时候，首先要调查一下看能不能使用已有的控件。借用已有控件可以大大节省我们自己的开发时间了。我们这个游戏界面上的按钮、文本框等自然可以使用 LabVIEW 自带的控件；黑白棋的棋盘棋子，也可以上网去找找看有没有别人已经做好的可供使用。

假如没有现成的棋盘棋子控件，那就要我们自己来做一个了。虽然作为整体，没有现成的东西可用，但把它细分成小的基础部分，还是有可能利用一些已有控件的。

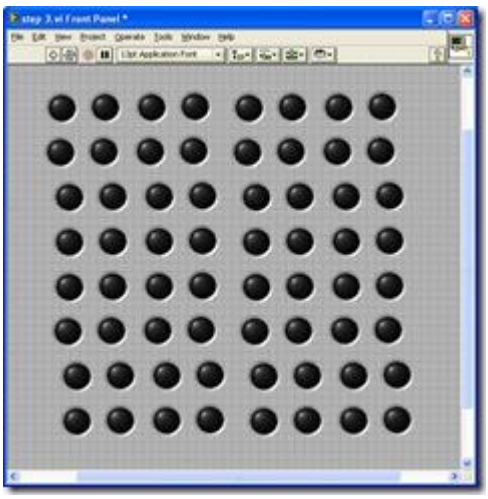
比如说棋子：这个游戏的棋子为圆形，只有黑白两色，个数最多 64 个。这个特点很适合用 LabVIEW 中的圆形 LED 灯泡来表示。圆形 LED 灯泡控件如下图所示：



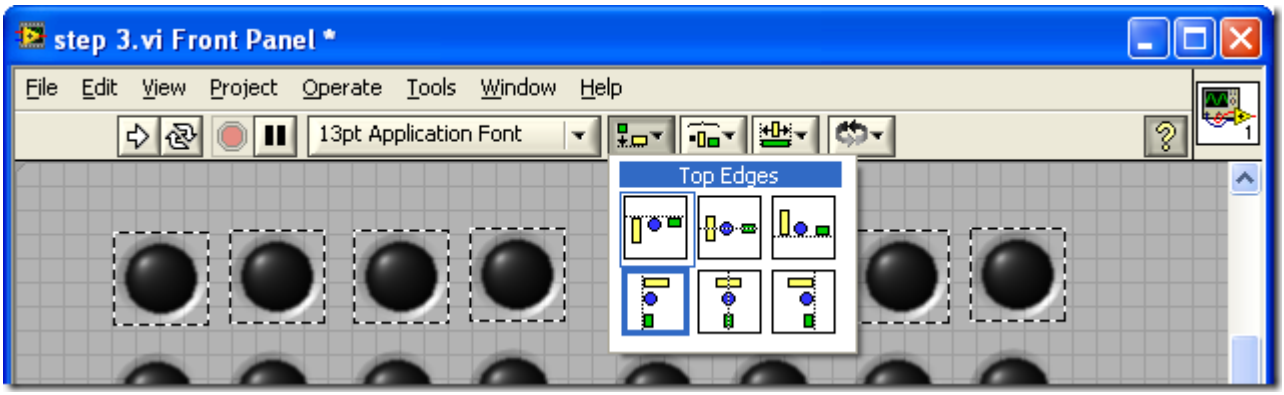
为了使它更像棋子，我们还要对他进行一下加工。首先，要把它的尺寸调大；用工具选板上的颜色画笔工具把它在“真”“假”状态下的颜色分别设置成黑色和白色；给他起一个有意义的名称–chess 0，但是在前面板上需要把这个标签隐藏起来，这个名声是为了以后编程的。改进后的棋子，如下图所示：



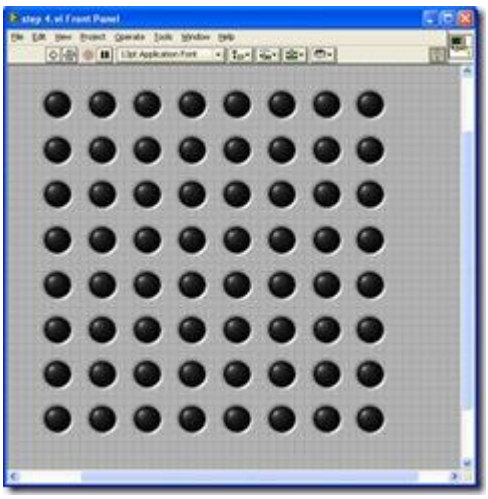
我们总共需要 64 个这样的棋子，排成 8 行 8 列。其它的棋子不需要再一个一个添加，以第一个棋子为模板，拷贝复制，就生成了第二个；再把两个棋子都选中，复制生成四个；重复这一过程，生成 8、16、32、64 个棋子。如下图所示：



下面我们要把这些棋子排列整齐。如果有耐心，可以用鼠标一个一个的调整每个棋子的位置。LabVIEW 提供了几个小工具来帮我们整理界面控件的位置和大小，它们就是工具条上，字体调整按钮右侧的四个按钮。这四个个按钮分别用于对齐控件，调整控件间距调整控件大小和控件前后次序。这几个工具在编辑界面时会经常使用到。



我们先把首先利用对齐工具把首行和首列棋子对齐、再利用间距调整按钮使它们间距均匀。再利用对齐工具让其它棋子都与首行首列对齐即可。调整好的界面如下：



到此为止，棋子的界面部分就完全设计好了。但是我们还要考虑一下相关的代码。棋子在程序运行过程中时发生变化的。

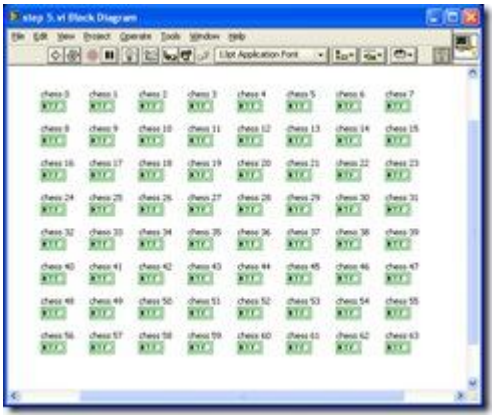
64 颗棋子并不都是显示在屏幕上的。游戏一开始，屏幕上只有四颗棋子，以后每走一步多一颗棋子。LabVIEW 每个控件都有一个属性“Visible”，控制控件是否在前面板上显示出来。棋盘的某个位置还没有放棋子时，可将该位置的棋子控件隐藏。

设计界面时，经常遇到有些控件只在某种特定情况下出现。这样的问题有两种最常见解决方案，一是我们刚刚提到的，可以在不需要看见某个控件时设置它的 Visible 属性，将其隐藏。这种方法代码编写比较简单，但是不利于界面编辑。尤其当界面某一位置需要在不同情况下出现多种不同控件的情况下。几个几个控件需要在那个位置上重叠摆放，不利于对控件进行编辑调整。

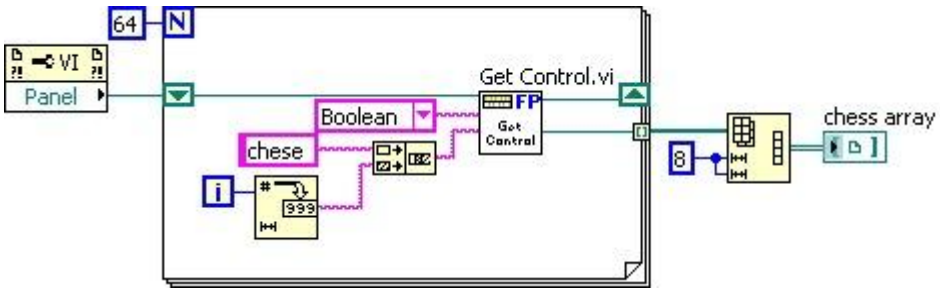
第二种方法是通过控件的 Position 属性，设置它在界面上的位置。需要显示控件时，把它设定到应该出现的位置；需要隐藏它的时候，把它挪到 VI 前面板可视范围之外的某个位置上，这样就看不到它了。使用这种方法，始终可以在 VI 前面板上找到这个控件进行编辑修改。但是编程的时候相对繁琐，需要在程序中设定控件的位置。

如果有一组控件需要同时出现或隐藏，那还可以考虑利用 tab 控件。把这组控件加在 tab 的某个页面上，然后通过调整 tab 的显示页面，控制控件出现与否。

打开程序的框图，64 个控件端子排布在那里。对它们分别进行操作，程序代码将会非常杂乱难懂。为了让程序更清晰，最好把这 64 个控件按照在棋盘上的位置，组织成一个 8×8 的二维数组。之后，程序对哪个位置的棋子进行操作就一目了然了。



直接把它们组成数组的方法是：为每个控件建立一个引用，然后使用 build array 函数把它们组织起来。但是对 64 个控件进行一一操作还是够烦的，最好可以编程解决。由于这 64 个棋子的名字是有规律的，因此我们可编程，按照名字一一等到这些控件的引用。再将得到的引用转换成 8×8 的数组。如下图所示的代码

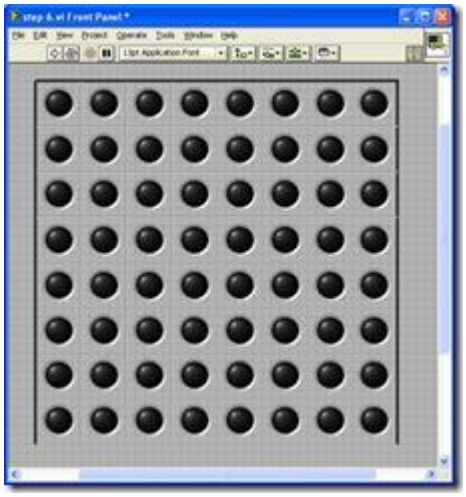


这里使用了一个关键的子 VI，Get Control.vi。这是 LabVIEW 自带的一个 VI（[LabVIEW]\resource\importtools\Common\VI Scripting\VI\Front Panel\Method\Get Control.vi），它用来按名称得到前面板上控件的引用。

这段代码输出的 chess array 是一个 8×8 数组，包含了所有 64 个控件。之后程序再对棋子进行操作，从这里得到相应位置的棋子的引用即可对其进行操作了。

实际工作中，有些应用程序有比较复杂的界面，为了简化它的代码，对界面控件的操作被放置在子 VI 中完成。直观的做法也是：程序开始时为主程序的控件建立引用，把这些引用捆绑成一个簇，传递到子 VI 中去。但是，一旦界面发生变化，所有使用到这个簇的 VI 都可能需要被修改，相当不便。所以，这样的程序也可以使用上段文字介绍的方案，只把主 VI 的引用传递给子 VI，在使用到某个主 VI 控件的时候，按照名字得到它的引用再对其进行操作。

现在棋子都已经摆放到位了，下面考虑如何把棋盘加上去。由于棋盘是静态不动的，所以设计起来要比棋子简单。LabVIEW 自带了这种形状的装饰组件，比如线条、方块之类的，利用这些装饰图案，很容易搭出一个棋盘来。如下图，就是由几根被画成黑色的线条搭出来的部分棋盘。

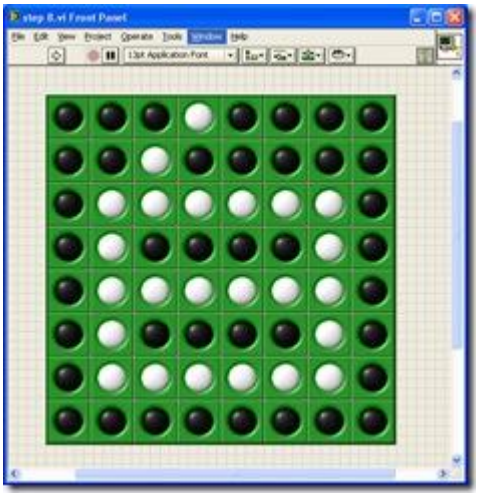


在编写程序界面时，装饰最常被用来将一组相关联的控件包围起来，或把不相关的控件个离开。

不过呢，用 LabVIEW 自带的简单图形拼出来的棋盘始终是不够漂亮。我们可以先用专业的画图工具，比如画图板（也不怎么专业吗）画一个漂亮的棋盘，保存成图片文件。然后把图片贴到 VI 的前面板，当作背景图片。这样，就可以得到一个漂亮的多的棋盘了。

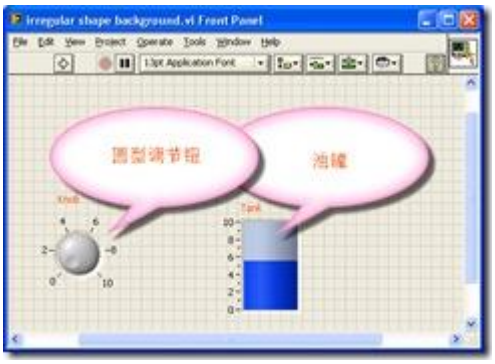
贴图这个操作，可以用 Ctrl+C，Ctrl+V，也可以直接在文件浏览器中，用鼠标把图片文件拖拽到 VI 前面板。

拖拽到 VI 上的图片，是处在界面最上层的，覆盖住了棋子。利用 Reorder 工具中的“Move to Back”，把它挪到最下层。在调整好棋子和棋盘的位置，整个一个棋子棋盘界面就做好了。如下图：

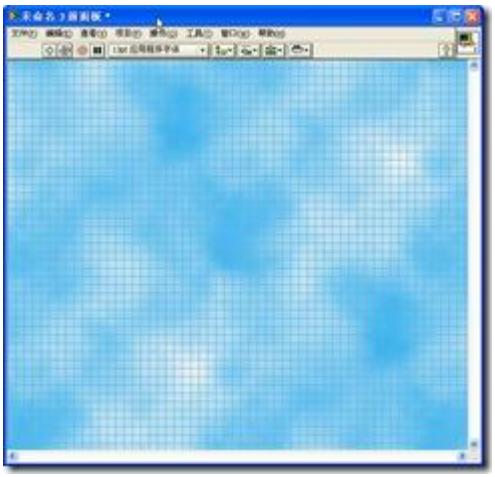


棋盘棋子都摆好之后，它们的相对位置应该固定下来。如果需要它在界面上挪动，应该是所有的棋子和棋盘一起动。先用鼠标把棋盘和全部棋子选中，再在 Reorder 工具中，选择 group 就可以把它们设定为一组。之后他们之间的相对位置就固定下来了。

咱们刚刚贴上来的图片是个矩形的，可是有时候，需要背景图片是不规则形状的。这种情况下需要使用支持透明色的图片格式，比如 gif 格式，把不规则图片空白部分设为透明即可。还有一种常用的文件格式 png 格式，支持像素点透明度的设置，利用不同的透明度设置还可以给背景图片做出阴影等效果。例如下图 VI 界面中两个带粉色的带阴影效果的解说框，使用的就是 png 文件格式的图片。



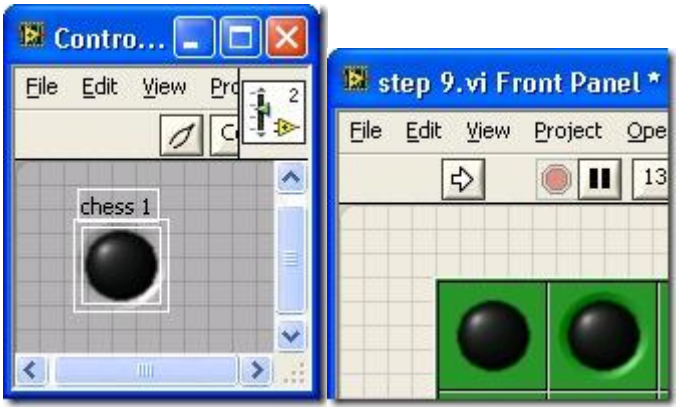
这样子贴上来的背景图片大小尺寸是固定的。有时候我们希望整个前面板被某一背景图片铺满。这个设置是在窗格的属性中设定的。鼠标右键点击前面板窗口的滚动条，选择“属性”，在“背景”设置中选择一个背景，或添加一幅自己的图片，就可以为 VI 前面板设置上背景图片了。下图是使用的“Clouds”背景的效果：



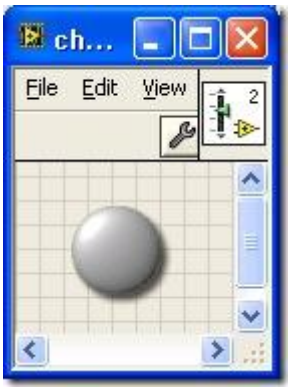
现在棋盘已经很漂亮了，相对来说，棋子似乎残留了比较多的 LabVIEW 控件的痕迹，与我们要编写的游戏程序风格不符。我们可以利用用户自定义控件来造出更漂亮的棋子。

制作自定义控件，可以从一个全新的 ctl 文件开始，也可以在某个已有的控件基础上进行修改。比如这里我们想把棋子周围一圈光效移走，右键点击一颗棋子，选择 Advanced–>Customized，弹出控件编辑界面。按工具栏上扳手一样的按钮，切换到自定义模式，即可修改控件上的元素。这个控件

有三个元素：标签、灯泡的主体部分、和边框。选中最外面那个白色的框，即边框，删除即可。编辑完成保存，新的棋子就不再有边框了。



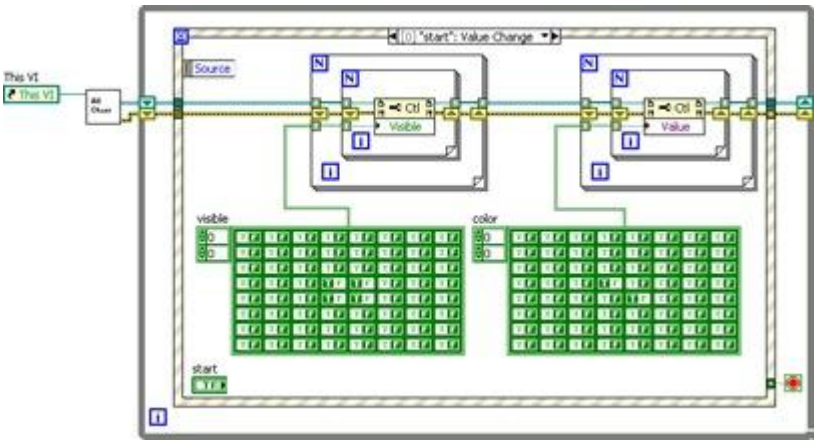
自定义控件也可以贴图，布尔型控件，比如按钮一般有 4 个状态，可以贴上 4 张不同的图片，做成复杂形状的按钮。下图就是通过贴图做成的一个有阴影效果的棋子按钮。

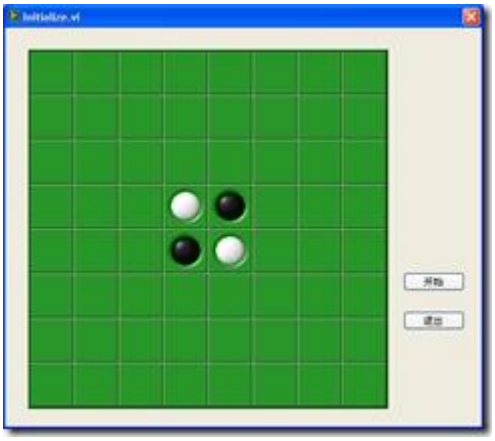


最好所有的控件都使用严格类型定义，这样以后再需要改变界面的时候，只要在类型定义 ctl 文件中改动，所有的棋子就都会改变。

到目前为止，棋盘棋子的界面已经基本成型。下面我们实现一小部分代码，来看看这个界面设计方案是否可行，是否可以改进。

以棋盘的初始化为例，在游戏开始时，只有两黑两白四颗子，摆在期盼最中间。实现这个操作的代码和执行结果如下图所示：





代码中的子 VI（Get All Chess.vi）中的代码，就是我们在第一节图 8 中看到的那段代码。它负责得到所有棋子的引用，并排列成二维数组。

这段初始化的代码并不算复杂，但是我们还是可以从中看出一些问题：棋子的布局需要用两个数组才可表达清楚，这给编程增加了负担。造成这一状况的根本原因在于：每个位置上的棋子实际上有三个状态：黑、白、无；而我们选用的灯泡控件，只有两个值：真、假。用这两个值不足以完全表达棋子的状态，所以，要两个布尔类型才能确定一个棋子的状态。

另外，棋盘只是一张背景图片，这样，判断鼠标在棋盘哪个位置上进行的点击，也比较繁琐。一旦棋盘挪动，代码也需要做相应改动。

改进的方法，就是使用一个有更多值的控件来表示棋子。在我们的程序中，要求每个棋子位置有多个不同图片，这正好可以使用 Pict Ring 控件。Pict Ring 控件包含三个值：空白图片、黑色棋子图

片、白色棋子图片。这样即便是没有棋子的位置，Pict Ring 控件还在，可以感知用户的鼠标点击事件。

上面的代码还有一处不足，每个棋子都是一个独立的控件，造成界面控件太多，不好管理。对于更复杂的程序，比如围棋游戏，如果使用这种方式，界面上就有将近 400 个控件，这是不可接受的。

改进的方法是使用数组控件，把所有的棋子组成一个数组。

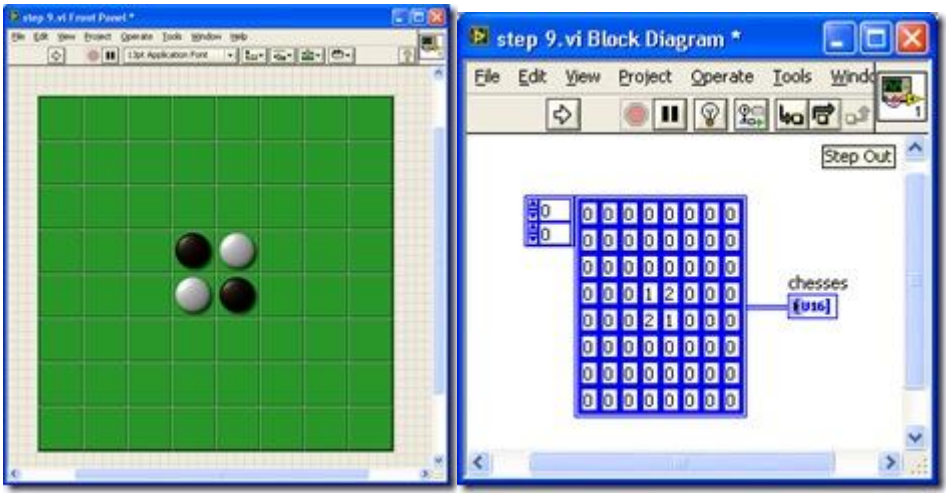
具体的实现步骤如下：

首先创建一个经典风格的 Pict Ring 控件，添加三幅图片：空白、黑棋子和白棋子。棋子采用的都是 png 文件格式的图片，以实现透明和阴影效果。

在程序中，我们不希望看见 Pict Ring 控件的边框和背景。我们可以用透明画笔，把边框和背景画为透明。

造一个二维数组用来放置棋子元素。由于界面上不希望看到这个数组的边框和背景，所以同样用透明画笔把它们画为透明。数组的标签、索引显示可以通过数组的右键菜单->显示一项进行隐藏。

把棋子元素放置于数组中，并把数组拉成 8×8 大小，放置在棋盘上放。一个棋盘棋子控件就做好了。而这种做法又大大简化了编码的复杂度，比如同样是初始化设置，只要一个赋值语句就可完成：



有时候需要画一个比较复杂图形或曲线，而 LabVIEW 没有提供相应的控件。可以借用 LabVIEW 已有的基本功能的控件，配上一些代码，实现一个具有特定功能的控件。

常被用来做这种基本控件有 XY Graph、3D Picture Control、Picture 控件等。

例如需要做一个绘制极坐标函数曲线的控件，就可以在 XY Graph 的基础上改造。一共一个转换用的 VI，把点的极坐标转换成直角坐标系下的值，在 XY Graph 上绘制出来就可以了。需要某个支持某种特定三维绘图方式的控件，可以通过改造 3D Picture Control 得到。

Picture 控件是个更为基础的控件，很多具有特殊效果的界面元素都可以利用 Picture 控件制作。比如，需要制作带图标的菜单，或类似 LabVIEW 函数选板的菜单等。LabVIEW 没有为它们提供现成的控件，就可以在 Picture 控件上自己把这些效果都画出来。我们前面介绍的棋盘棋子界面也可以使用 Picture 控件来制作。

下面介绍一下实现这个界面的具体过程。

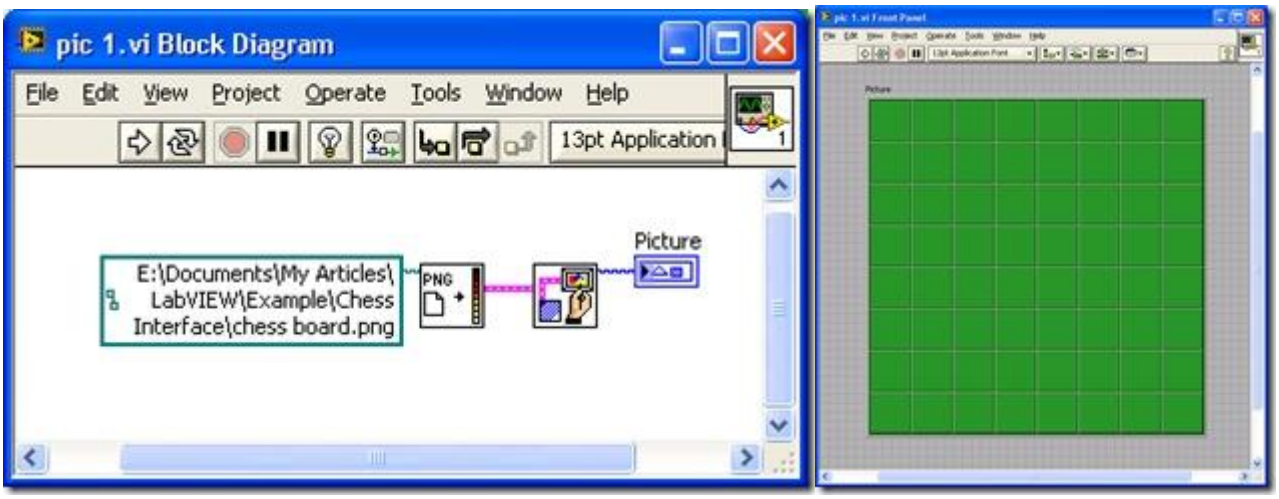
第一步，创建一个空白的 Picture 控件，针对 Picture 控件的常用操作都在 Picture Functions 函数选板中。

与前面介绍的方法不同，使用 Picture 控件制作棋盘棋子的过程，不是在 VI 编辑状态下进行的，而是需要在程序运行时绘制。所以下面的界面设计工作都要通过编程来完成了。先介绍一下 Picture 控件的 Erase First 属性，它有 3 个值：0 表示从不擦除，也就是说每次传一个数据给这个控件，比如一个圆环图案，Picture 上显示的并非只有这个圆环，而是把圆环叠加在原本的内容之上。如果我想画一个有三个矩形组成的图案，可以分三次画，每一次传递一个矩形图案给 Picture 控件；2 表示每

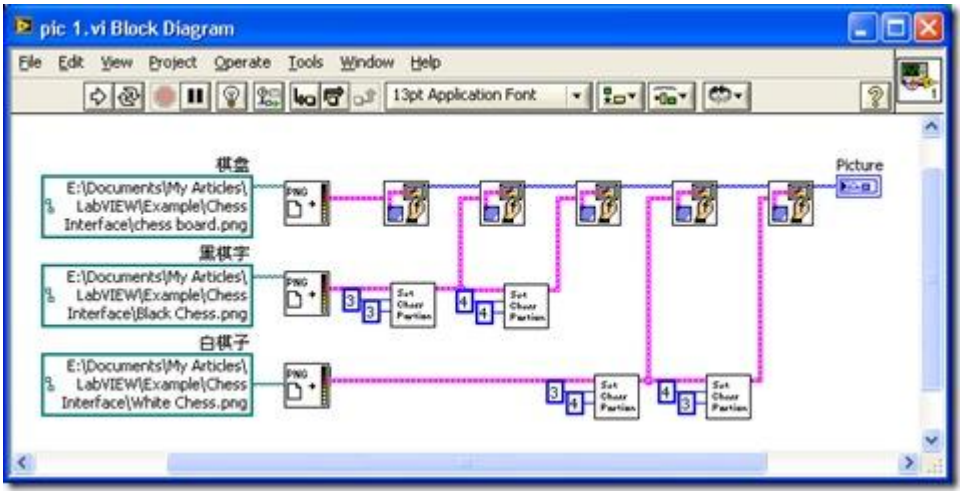
次都擦除，每次传递一个图案给 Picture 控件，它都会将原来的图案擦掉，仅保留这一次的图形。擦除图案后 Picture 控件会显示默认的白色。所以，使用这种方式，用户在切换图案时会看到 Picture 闪烁一下。若非必要，尽量不要使用这种方式；1 表示程序第一次运行时把 Picture 上的内容清除，等于自动帮你做了初始化工作，我们这里也使用这种方式。

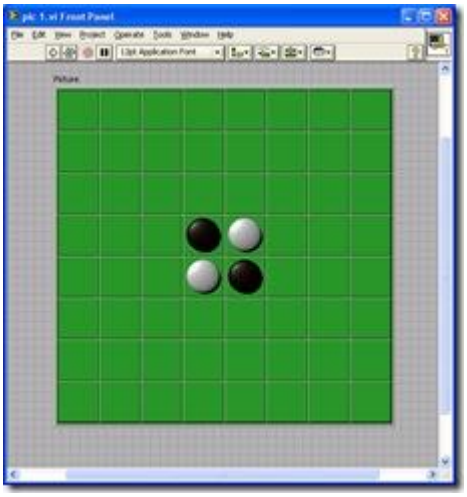
就是说，棋盘布局发生变化时，进更新发生了变动的位置。不要重绘整幅图。

棋盘再棋子下层，所以要先画棋盘。画棋盘可以使用 LabVIEW 提供的划线函数，一条线一条线画出来。因为我们之前已经制作了棋盘的图片，所以可以直接把这张图片显示出来。代码如下：



下面再画上棋盘初始时的四个棋子。画棋子的方法与棋盘相同，可以使用画圆函数，已可以使用已经制作好的图片：





到目前为止，界面设计的几种方法就已经介绍好了。如果能够把这个黑白棋的相关界面和操作（比如放置棋子，反转棋子等）提取出来，合成一个组件，公布出来，其他有类似需求的人就可以直接利用这个组件，不再需要自己重新设计了。

然而，在 LabVIEW 8 之前是无法实现这一功能呢，因为控制棋子行为的代码分散在程序的各处，而棋盘棋子也是主 VI 的一部分，很难将它们提取出来组成独立模块。LabVIEW 8 中出现的 XControl 可以把控件的界面及行为封装在一起，成为一个既有界面，又有运行代码一个组件。