

LabVIEW Draw Revealed

Copyright © 2002 by the McGraw-Hill Companies, Inc. All rights reserved.
Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without prior written permission of the publisher.

LabVIEW Draw Revealed

Contents

1.1 INTRODUCTION.....	3
1.2 THE LABVIEW DRAW USER INTERFACE.....	4
1.3 THE LABVIEW DRAW VI DIAGRAMS.....	4
1.3.1 THE DRAWING FUNCTIONS	6
1.3.2 THE FILE SAVE AND EXPORT FUNCTIONS	6

LabVIEW Draw Revealed

1.1 Introduction

LabVIEW video games are a great source of advanced Picture control techniques, but the mathematics and the diagram complexity necessary to realize a video game can be overwhelming for the uninitiated. In order to explore Picture control techniques in a practical, yet accessible setting, this article introduces *LabVIEW Draw*, a simple drawing VI that takes advantage of the native LabVIEW Picture functions to simulate a simple graphics program (see Figure 1). In order to make the VI more interesting, load, save, import and export capabilities have been added so users can save and load images in a variety of formats. In addition, an implementation of the undo function introduced in Chapter 7 of the text has been added to the VI to enable drawing error corrections.

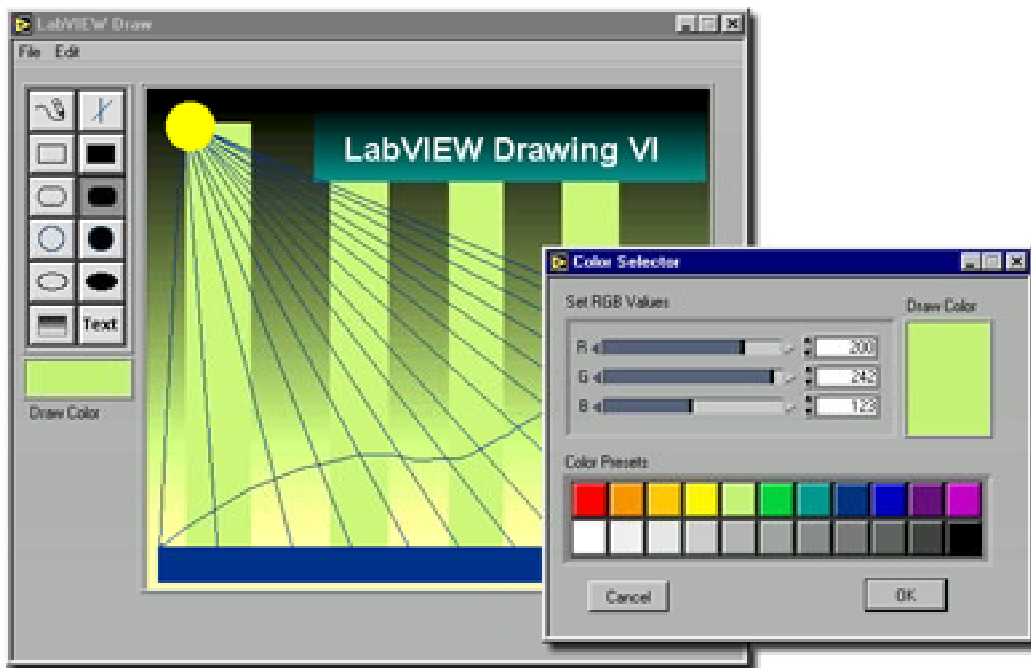


Figure 1 The LabVIEW Draw GUI Panels

1.2 The LabVIEW Draw User Interface

The primary LabVIEW Draw GUI panels are revealed in Figure 1 (above). Notice that the pop-up Color Selector subVI panel is superimposed over the main GUI panel in this depiction. The Color Selector subVI permits the user to select colors from a bank of presets, or to set the R, G, and B values directly. The Color Selector subVI is a non-modal GUI VI that uses the “Selective Panel Control” technique introduced in Chapter 8 of the text. This non-modal design permits the Color Selector to remain open and active while the user is drawing with the various tools on the main GUI panel. The user can interactively select and try colors without closing the pop-up subVI. This provides immediate access to color editing while constructing an image.

Drawing takes place on the main GUI panel inside a large Picture control. The user selects a drawing tool from the toolbar on the left, then uses the selected tool to draw a shape or line inside the Picture control. Drawing tools include a freeform pencil, a line tool, filled and unfilled primitive shapes (rectangles, circles, rounded rectangles and ovals), a gradient tool, and a text tool. The selected shape is rendered using the currently selected color. While the toolbar includes no eraser tool, 5 undo levels permit the most recent actions to be erased. This mechanism is a practical application of the multiple undo strategy introduced in Chapter 7 of the text.

1.3 The LabVIEW Draw VI Diagrams

The main VI diagram is depicted in Figure 2. The uppermost subVI in the left-hand corner (outside the main while loop) generates a custom menu for the VI. Image “Load”, “Save”, “Import” and “Export” functions are accessible from the menus, along with “Clear Image”, “Undo”, and “Redo” functions. Just below the Menu Builder subVI, the Undo manager subVI is initialized. As you may recall from “multiple undo” example presented in Chapter 7 of the text, an array of undo and redo states is stored inside the shift register of the encapsulated Undo subVI. Before the Undo array can be used, it must first be initialized. Five copies of an empty Picture control constant are stored in the Undo array during initialization. Below the Undo initialization, a value of zero is passed to a shift register counter. This counter controls the sequential execution of multi-step drawing operations. (A more detailed examination of this shift register counter is presented below.)

Moving inside the main while loop, the standard menu functions in the upper left corner are responsible for enabling and disabling the Undo and Redo functions, and for reading user selections. Menu processing actually takes place in the smaller case structure on the right-hand side of the loop. Undo/redo processing and file operations are located in this case structure (labeled “Menu Functions” in the figure). These operations are the last step in the chain before the front panel Picture indicator is updated.

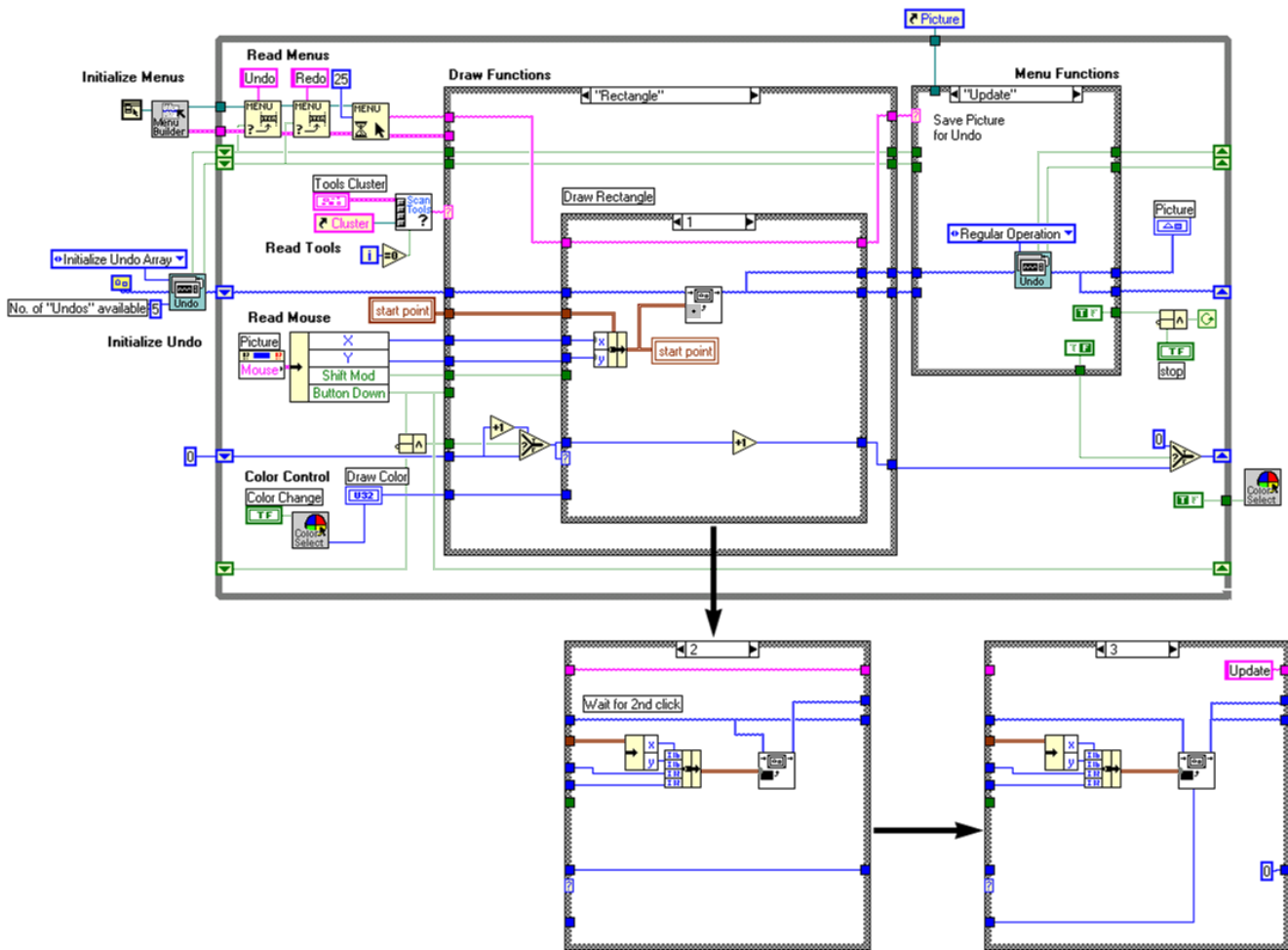


Figure 2 LabVIEW Draw – Main VI diagram and highlights

Below the logic responsible for reading the menus is the logic responsible for reading the Tools cluster. The “Scan Tools” subVI monitors the Toolbar for changes, and controls tool selection by passing the appropriate string to the Draw Functions case structure. Below the Toolbar logic is the key to the drawing operations. The mouse position and the state of the mouse buttons are polled using the Picture control’s mouse property. These are used to orchestrate the drawing operations - as we shall see in a moment.

The subVI in the bottom left-hand corner is the Color Selector subVI. As noted previously, this subVI is “GUI-enabled” using Selective Panel Control. The Color Selector subVI executes on every loop iteration, and the current draw color originates inside this subVI whether its front panel is open or closed. When the “Color Change” button is clicked on the main VI panel, this subVI uses VI Server functions to open its own panel for user interaction. When the user clicks OK or Cancel, this subVI uses VI Server to close itself. The important thing to notice is that regardless of whether its front panel is open or closed, this subVI remains active in the execution chain.

1.3.1 The Drawing Functions

The diagram elements responsible for the drawing functions are:

- The Tools Cluster
- The Picture indicator's "Mouse" properties
- The value in the draw counter shift register

The Tools cluster determines which of several draw sequences is currently selected - the filled rectangle draw sequence is featured in the figure, and therefore will be the basis of this discussion.

When the VI is started, the draw counter shift register is initialized to zero. The first mouse click increments the draw counter to 1 and initiates the draw sequence. When the mouse button is clicked for the first time, the current x and y mouse coordinates are passed to the "Draw Point" function to draw an initial point. At the same time, these initial coordinates are stored in "start point" for later use, and the draw counter is incremented to 2 to activate mouse tracking (see Draw Rectangle – Case number 1 in the figure). Case number 2 (situated directly below the main while loop in the figure) illustrates active mouse tracking. As the user moves the mouse around inside the picture control, the "Draw Rectangle" function dynamically generates a rectangle based on the current mouse position. Bounded by the initial point at one corner and the current cursor position at the opposite corner, the size of the rectangle continues to track the moving mouse. This tracking behavior continues until the user clicks the mouse button for a second time. When the second click is detected, the draw counter is incremented to 3, and the shape of the rectangle is locked in. The complete rectangle is drawn from the initial point (captured with the first mouse click and stored in "start point") to the current mouse position as depicted in case number 3 (below the main loop). After the final rectangle is drawn and appended to the previous image, the final step is to reset the draw counter to zero in preparation for the next draw sequence. While different drawing tools use different variations of this sequence, the technique is essentially the same for all tools.

1.3.2 The File Save and Export Functions

While LabVIEW Draw was designed primarily to demonstrate LabVIEW's built-in Picture functions, this VI also offers a few interesting tricks for saving and loading graphic files. Techniques for loading images into a Picture indicator were covered previously in the text of Chapter 9, but saving images to disk has not been covered. And while there are VIs specifically devoted to both loading and saving images in standard formats, the first step involved in saving a Picture control image can be somewhat elusive.

In order to save the current Picture control image, you must invoke the "Get Image" method. Similar in function to the "Get Panel Image" method accessed from a VI Server Invoke node, the "Get Image" method is accessed through a Picture control Invoke node. You can create an Invoke node for a control or indicator in much the same way as you create a control Property

node – just pop-up on the control or diagram terminal and select “Invoke Node” from the “Create” menu. Alternatively, you can create a reference to the Picture control, and wire this reference into a standard Invoke node from the “Application Control” palette. Either way, you will have access to the “Get Image” method through the Picture control’s Invoke node. Once you have selected the “Get Image” method, the inputs and outputs presented on the node are reasonable self-explanatory. If you have difficulty understanding any of the parameters, the online help (accessed by popping-up on the node) offers additional information about each item.

All image file “Save” and “Export” functions in the LabVIEW Draw VI use the file save subVI depicted in Figure 3. When this subVI is called, the main VI passes a reference to its Picture indicator. This reference is used to invoke the “Get Image” method inside the subVI. The flattened image, color table, bit depth and bounds of the image are then passed to the appropriate file write subVI, and the image is written to disk in the designated file format.

A copy of LabVIEW Draw has been included on the CD-ROM. You are invited to experiment with it and observe the techniques used. Particularly interesting are the Undo implementation and the gradient fill tool. If you are so inclined, you may want to add new functionality to this VI, such as an eraser tool, a fill bucket, variable line thickness, and so on. If you come up with something exciting and want to share it with others, email a copy of your solution to the author – solutions@bettervi.com. Interesting submissions will be posted on the BetterVIEW website (bettervi.com).

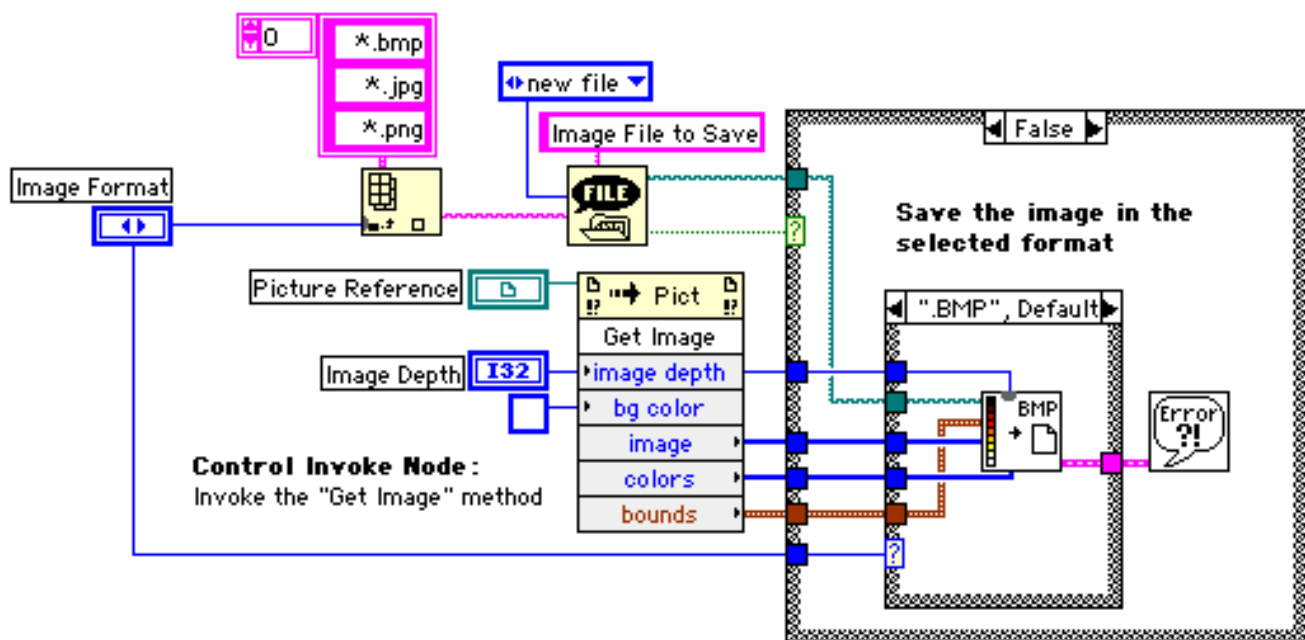


Figure 3 The subVI responsible for saving and exporting LabVIEW Draw images.