Magic Tutorial #5: Multiple Windows

Robert N. Mayo

Computer Science Division
Electrical Engineering and Computer Sciences
University of California
Berkeley, CA 94720

(Updated by others, too.)

This tutorial corresponds to Magic version 6.

Tutorials to read first:

Magic Tutorial #1: Getting Started
Magic Tutorial #2: Basic Painting and Selection

Commands introduced in this tutorial:

:center :closewindow, :openwindow, :over, :specialopen, :under, :windowpositions

Macros introduced in this tutorial:

o, O, ‘‘,’’

1. Introduction

A window is a rectangular viewport. You can think of it as a magnifying glass that may be moved around on your chip. Magic initially displays a single window on the screen. This tutorial will show you how to create new windows and how to move old ones around. Multiple windows allow you to view several portions of a circuit at the same time, or even portions of different circuits.

Some operations are easier with multiple windows. For example, let’s say that you want to paint a very long line, say 3 units by 800 units. With a single window it is hard to align the box accurately since the magnification is not great enough. With multiple windows, one window can show the big picture while other windows show magnified views of the areas where the box needs to be aligned. The box can then be positioned accurately in these magnified windows.

2. Manipulating Windows

2.1. Opening and Closing Windows

Initially Magic displays one large window. The

:openwindow [*cellname*]

command opens another window and loads the given cell. To give this a try, start up Magic with the command **magic tut5a**. Then point anywhere in a Magic window and type the command **:openwindow tut5b** (make sure you're pointing to a Magic window). A new window will appear and it will contain the cell **tut5b**. If you don't give a *cellname* argument to **:openwindow**, it will open a new window on the cell containing the box, and will zoom in on the box. The macro **o** is predefined to **:openwindow**. Try this out by placing the box around an area of **tut5b** and then typing **o**. Another window will appear. You now have three windows, all of which display pieces of layout. There are other kinds of windows in Magic besides layout windows: you'll learn about them later. Magic doesn't care how many windows you have (within reason) nor how they overlap.

To get rid of a window, point to it and type

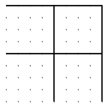
:closewindow

or use the macro **O**. Point to a portion of the original window and close it.

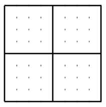
2.2. Resizing and Moving Windows

If you have been experimenting with Magic while reading this you will have noticed that windows opened by **:openwindow** are all the same size. If you'd prefer a different arrangement you can resize your windows or move them around on the screen. The techniques used for this are different, however, depending on what kind of display you're using. If you are using a workstation, then you are also running a window system such as X11 or SunView. In this case Magic's windows are moved and resized just like the other windows you have displayed, and you can skip the rest of this section.

For displays like the AED family, which don't have a built-in window package, Magic implements its own window manager. To re-arrange windows on the screen you can use techniques similar to those you learned for moving the box for painting operations. Point somewhere in the border area of a window, except for the lower left corner, and press and hold the right button. The cursor will change to a shape like this:



This indicates that you have hold of the upper right corner of the window. Point to a new location for this corner and release the button. The window will change shape so that the corner moves. Now point to the border area and press and hold the left button. The cursor will now look like:



This indicates that you have hold of the entire window by its lower left window. Move the cursor and release the button. The window will move so that its lower left corner is where you pointed.

The other button commands for positioning the box by any of its corners also work for windows. Just remember to point to the border of a window before pushing the buttons.

The middle button can be used to grow a window up to full-screen size. To try this, click the middle button over the caption of the window. The window will now fill the entire screen. Click in the caption again and the window will shrink back to its former size.

2.3. Shuffling Windows

By now you know how to open, close, and resize windows. This is sufficient for most purposes, but sometimes you want to look at a window that is covered up by another window. The **:underneath** and **:over** commands help with this.

The **:underneath** command moves the window that you are pointing at underneath all of the other windows. The **:over** command moves the window on top of the rest. Create a few windows that overlap and then use these commands to move them around. You'll see that overlapping windows behave just like sheets of paper: the ones on top obscure portions of the ones underneath.

2.4. Scrolling Windows

Some of the windows have thick bars on the left and bottom borders. These are called *scroll bars,* and the slugs within them are called *elevators*. The size and position of an elevator indicates how much of the layout (or whatever is in the window) is currently visible. If an elevator fills its scroll bar, then all of the layout is visible in that window. If an elevator fills only a portion of the scroll bar, then only that portion of the layout is visible. The position of the elevator indicates which part is visible – if it is near the bottom, you are viewing the bottom part of the layout; if it is near the top, you are viewing the top part of the layout. There are scroll bars for both the vertical direction (the left scroll bar) and the horizontal direction (the bottom scroll bar).

Besides indicating how much is visible, the scroll bars can be used to change the view of the window. Clicking the middle mouse button in a scroll bar moves the elevator to that position. For example, if you are viewing the lower half of a chip (elevator near the bottom) and you click the middle button near the top of the scroll bar, the elevator will move up to that position and you will be viewing the top part of your chip. The little squares with arrows in them at the ends of the scroll bars will scroll the view by one screenful when the middle button is clicked on them. They are useful when you want to move exactly one screenful. The **:scroll** command can also be used to scroll the view (though we don't think it's as easy to use as the scroll bars). See the man page for information on it.

If you only want to make a small adjustment in a window's view, you can use the command

:center

It will move the view in the window so that the point that used to be underneath the cursor is now in the middle of the window. The macro **,** is predefined to **:center**.

The bull's-eye in the lower left corner of a window is used to zoom the view in and out. Clicking the left mouse button zooms the view out by a factor of 2, and clicking the right mouse button zooms in by a factor of 2. Clicking the middle button here makes everything in the window visible and is equivalent to the **:view** command.

2.5. Saving Window Configurations

After setting up a bunch of windows you may want to save the configuration (for example, you may be partial to a set of 3 non-overlapping windows). To do this, type:

:windowpositions *filename*

A set of commands will be written to the file. This file can be used with the **:source** command to recreate the window configuration later. (However, this only works well if you stay on the same kind of display; if you create a file under X11 and then **:source** it under SunView, you might not get the same positions since the coordinate systems may vary.)

3. How Commands Work Inside of Windows

Each window has a caption at the top. Here is an example:

mychip EDITING shiftcell

This indicates that the window contains the root cell **mychip**, and that a subcell of it called **shiftcell** is being edited. You may remember from the Tutorial #4 that at any given time Magic is editing exactly one cell. If the edit cell is in another window then the caption on this window will read:

mychip [NOT BEING EDITED]

Let's do an example to see how commands are executed within windows. Close any layout windows that you may have on the screen and open two new windows, each containing the cell **tut5a**. (Use the **:closewindow** and **:openwindow tut5a** commands to do this.) Try moving the box around in one of the windows. Notice that the box also moves in the other window. Windows containing the same root cell are equivalent as far as the box is concerned: if it appears in one it will appear in all, and it can be manipulated from them interchangeably. If you change **tut5a** by painting or erasing portions of it you will see the changes in both windows. This is because both windows are looking at the same thing: the cell **tut5a**. Go ahead and try some painting and erasing until you feel comfortable with it. Try positioning one corner of the box in one window and another corner in another window. You'll find it doesn't matter which window you point to, all Magic knows is that you are pointing to **tut5a**.

These windows are independent in some respects, however. For example, you may scroll one window around without affecting the other window. Use the scrollbars to give this a try. You can also expand and unexpand cells independently in different windows.

We have seen how Magic behaves when both windows view a single cell. What happens when windows view different cells? To try this out load **tut5b** into one of the windows (point to a window and type **:load tut5b**). You will see the captions on the windows change — only one window contains the cell currently being edited. The box cannot be positioned by placing one corner in one window and another corner in the other window because that doesn't really make sense (try it). However, the selection commands work between windows: you can select information in one window and then copy it into another (this only works if the window you're copying into contains the edit cell; if not, you'll have to use the **:edit** command first).

The operation of many Magic commands is dependent upon which window you are pointing at. If you are used to using Magic with only one window you may, at first, forget to point to the window that you want the operation performed upon. For instance, if there are several windows on the screen you will have to point to one before executing a command like **:grid** — otherwise you may not affect the window that you intended!

4. Special Windows

In addition to providing multiple windows on different areas of a layout, Magic provides several special types of windows that display things other than layouts. For example, there are special window types to edit netlists and to adjust the colors displayed on the screen. One of the special window types is described in the section below; others are described in the other tutorials. The

:specialopen *type* [*args*]

command is used to create these sorts of windows. The *type* argument tells what sort of window you want, and *args* describe what you want loaded into that window. The **:openwindow** *cellname* command is really just short for the command **:specialopen layout** *cellname*.

Each different type of window (layout, color, etc.) has its own command set. If you type **:help** in different window types, you'll see that the commands are different. Some of the commands, such as those to manipulate windows, are valid in all windows, but for other commands you must make sure you're pointing to the right kind of window or the command may be misinterpreted. For example, the **:extract** command means one thing in a layout window and something totally different in a netlist window.

5. Color Editing

Special windows of type **color** are used to edit the red, green, and blue intensities of the colors displayed on the screen. To create a color editing window, invoke the command

:specialopen color [*number*]

Number is optional; if present, it gives the octal value of the color number whose intensities are to be edited. If *number* isn't given, 0 is used. Try opening a color window on color 0.

A color editing window contains 6 ‘‘color bars’’, 12 ‘‘color pumps’’ (one on each side of each bar), plus a large rectangle at the top of the window that displays a swatch of the color being edited (called the ‘‘current color’’ from now on). The color bars display the components of the current color in two different ways. The three bars on the left display the current color in terms of its red, green, and blue intensities (these intensities are the values actually sent to the display). The three bars on the right display the current color in terms of hue, saturation, and value. Hue selects a color of the spectrum. Saturation indicates how diluted the color is (high saturation corresponds to a pure color, low saturation corresponds to a color that is diluted with gray, and a saturation of 0 results in gray regardless of hue). Value indicates the overall brightness (a value of 0 corresponds to black, regardless of hue or saturation).

There are several ways to modify the current color. First, try pressing any mouse button while the cursor is over one of the color bars. The length of the bar, and the current color, will be modified to reflect the mouse position. The color map in the display is also changed, so the colors will change everywhere on the screen that the current color is displayed. Color 0, which you should currently be editing, is the background color. You can also modify the current color by pressing a button while the cursor is over one of the ‘‘color pumps’’ next to the bars. If you button a pump with ‘‘+’’ in it, the value of the bar next to it will be incremented slightly, and if you button the ‘‘-’’ pump, the bar will be decremented slightly. The left button causes a change of about 1% in the value of the bar, and the right button will pump the bar up or down by about 5%. Try adjusting the bars by buttoning the bars and the pumps.

If you press a button while the cursor is over the current color box at the top of the window, one of two things will happen. In either case, nothing happens until you release the button. Before releasing the button, move the cursor so it is over a different color somewhere on the screen. If you pressed the left button, then when the button is released the color underneath the cursor becomes the new current color, and all future editing operations will affect this color. Try using this feature to modify the color used for window borders. If you pressed the right button, then when the button is released the value of the current color is copied from whatever color is present underneath the cursor.

There are only a few commands you can type in color windows, aside from those that are valid in all windows. The command

:color [*number*]

will change the current color to *number*. If no *number* is given, this command will print out the current color and its red, green, and blue intensities. The command

:save [*techStyle displayStyle monitorType*]

will save the current color map in a file named *techStyle*.*displayStyle*.*monitorType***.cmap**, where *techStyle* is the type of technology (e.g., **mos**), *displayStyle* is the kind of display specified by a **styletype** in the **style** section of a technology file (e.g., **7bit**), and *monitorType* is the type of the current monitor (e.g., **std**). If no arguments are given, the current technology style, display style, and monitor type are used. The command

:load [*techStyle displayStyle monitorType*] will load the color map from the file named *techStyle*.*displayStyle*.*monitorType***.cmap** as above. If no arguments are given, the current technology style, display style, and monitor type are used. When loading color maps, Magic looks first in the current directory, then in the system library.