Let Cell Maps and Traffic Lights Guide Your Spreadsheet Inspections

By Philip L. Bewig

April 6, 2004

How do you know your spreadsheet is right?

That is a very difficult question for spreadsheet developers to answer. Spreadsheets tend to grow haphazardly, without planning. They almost certainly evolve over time, and may have been modified by many different developers. And there is no shared culture among spreadsheet developers that requires testing before a spreadsheet is distributed, as there is among traditional computer programmers.

This note describes two tools that help spreadsheet developers understand the structures of their spreadsheets and provide a guide to cell-by-cell spreadsheet inspections, which is the best antidote to spreadsheet errors. Used conscientiously, these tools give spreadsheet developers confidence in their work, and convince a doubting user that the developer really does have reason for that confidence.

The tools work with Excel, and are distributed as Excel add-ins. CellMaps.xla and TrafficLights.xla are available from the author at pbewig@swbell.net. Copy these add-ins into your add-ins directory, then activate them on the Tools > Add-ins . . . menu. Be sure to deactivate the add-ins when you are finished using them, because they interact poorly with some other uses of spreadsheets, and so that they release their memory for other purposes. Also be sure to save your spreadsheet before using either of these tools, because both will destroy any interior colors you have specified in your spreadsheet; you can return to the original colors by simply reloading the original spreadsheet when you are finished with CellMaps and TrafficLights.

Cell Maps

The CellMap tool colorizes the cells of a spreadsheet according to the type of cell found. Input cells have dependents but no precedents; output cells have precedents but no dependents. Text cells and number cells contain literal text and numbers, respectively. Formula cells contain expressions introduced by an equal sign, and may be copied to the right or down. The CellMap toolbar is shown below:

Using the toolbar is simple: just click on the desired button to turn on the indicated colors, or click the white button to turn them all off. The two buttons at the right end of the toolbar turn precedent/dependent arrows on and off, and will be discussed momentarily.

We'll use an example to illustrate the use of the CellMap toolbar. The spreadsheet shown below calculates a five-year projection of sales and earnings for a new product that is being considered for distribution. Input data are the projected unit sales, selling price per unit, variable cost per unit, and fixed cost per year for the first year, as well as expected growth rates for each of those items, and a tax rate of 35% that applies only on positive earnings. Output data is after-tax net income each year for the next five years:

Microsoft Excel - FiveYearProjection.xls

File Edit View Insert Format Tools Data Window Help

R2C4 = = PriorYear * (1 + GrowthRate)

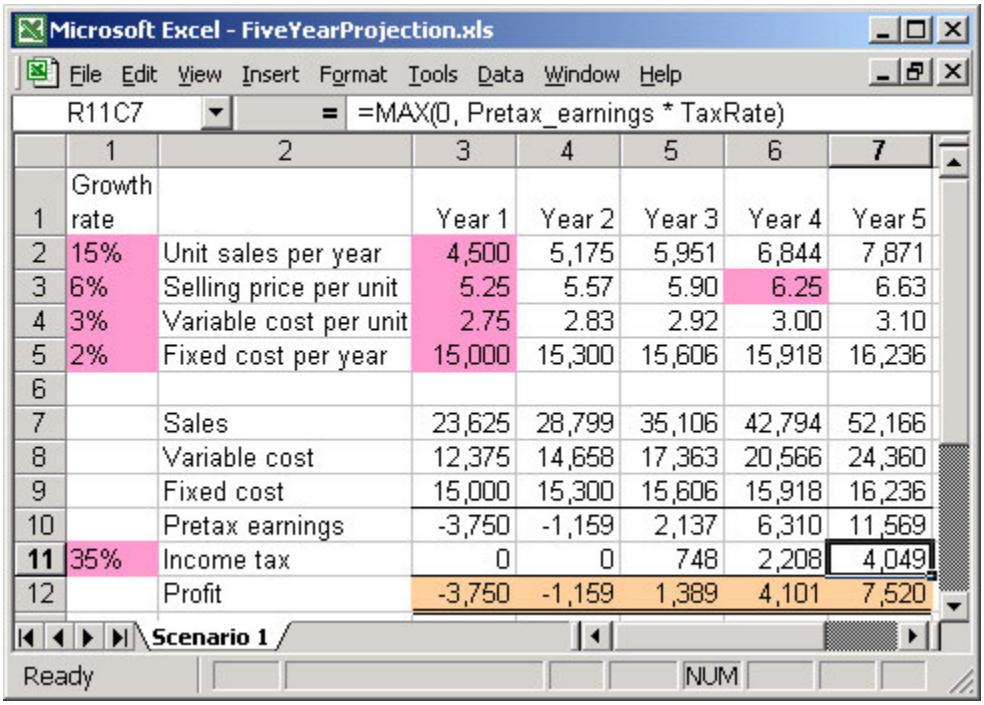
	1	2	3	4	5	6	7
1	Growth rate		Year 1	Year 2	Year 3	Year 4	Year 5
2	15%	Unit sales per year	4,500	5,175	5,951	6,844	7,871
3	6%	Selling price per unit	5.25	5.57	5.90	6.25	6.63
4	3%	Variable cost per unit	2.75	2.83	2.92	3.00	3.10
5	2%	Fixed cost per year	15,000	15,300	15,606	15,918	16,236
6							
7		Sales	23,625	28,799	35,106	42,794	52,166
8		Variable cost	12,375	14,658	17,363	20,566	24,360
9		Fixed cost	15,000	15,300	15,606	15,918	16,236
10		Pretax earnings	-3,750	-1,159	2,137	6,310	11,569
11	35%	Income tax	0	0	748	2,208	4,049
12		Profit	-3,750	-1,159	1,389	4,101	7,520

Scenario 1

Ready NUM

How would you check that spreadsheet to determine if it shows the correct answer? There are fifty-five cells containing numbers or formulas, presumably all of them either showing the final solution or somehow contributing to it. Where do you start?

Let's start by showing the spreadsheet with input cells colored rose and output cells colored tan, which is obtained by clicking the In and Out buttons on the CellMaps toolbar:



Microsoft Excel - FiveYearProjection.xls

File Edit View Insert Format Tools Data Window Help

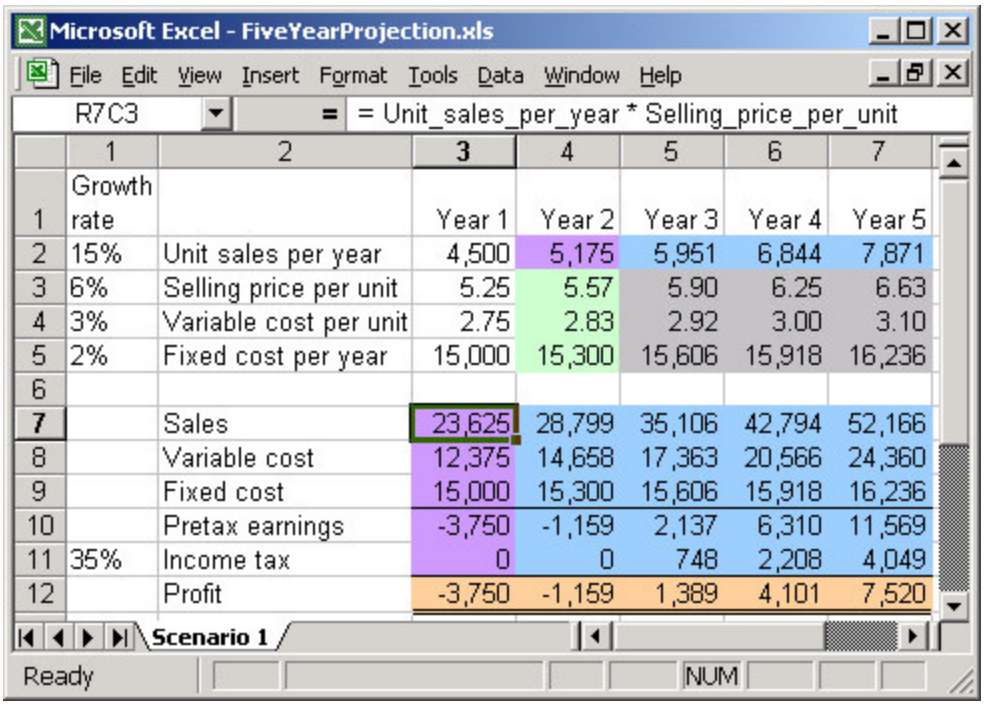
R11C7 = =MAX(0, Pretax_earnings * TaxRate)

	1	2	3	4	5	6	7
1	Growth rate		Year 1	Year 2	Year 3	Year 4	Year 5
2	15%	Unit sales per year	4,500	5,175	5,951	6,844	7,871
3	6%	Selling price per unit	5.25	5.57	5.90	6.25	6.63
4	3%	Variable cost per unit	2.75	2.83	2.92	3.00	3.10
5	2%	Fixed cost per year	15,000	15,300	15,606	15,918	16,236
6							
7		Sales	23,625	28,799	35,106	42,794	52,166
8		Variable cost	12,375	14,658	17,363	20,566	24,360
9		Fixed cost	15,000	15,300	15,606	15,918	16,236
10		Pretax earnings	-3,750	-1,159	2,137	6,310	11,569
11	35%	Income tax	0	0	748	2,208	4,049
12		Profit	-3,750	-1,159	1,389	4,101	7,520

Scenario 1

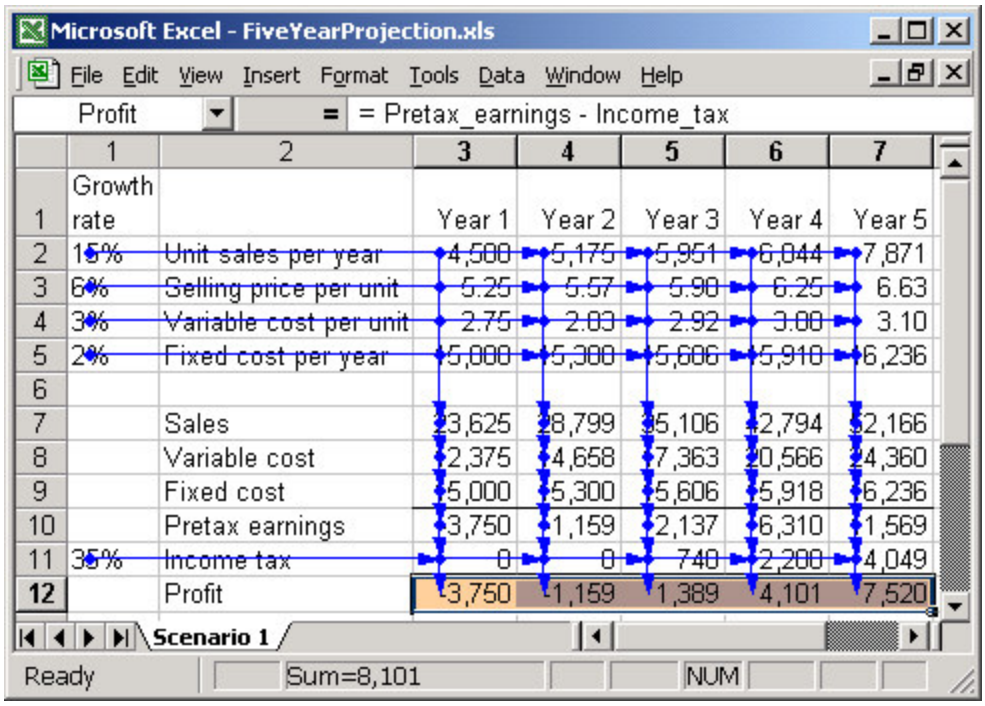
Ready NUM

This spreadsheet has nine input cells and five output cells; the remaining forty-one non-text cells contain intermediate formulas. Here is the same spreadsheet again, with those forty-one intermediate cells and the five output cells colored. The formula cells show the structure of the spreadsheet, with original formulas at R2C4 and R7:12C3 and copies elsewhere, with the various colors showing the direction from which each cell was copied:

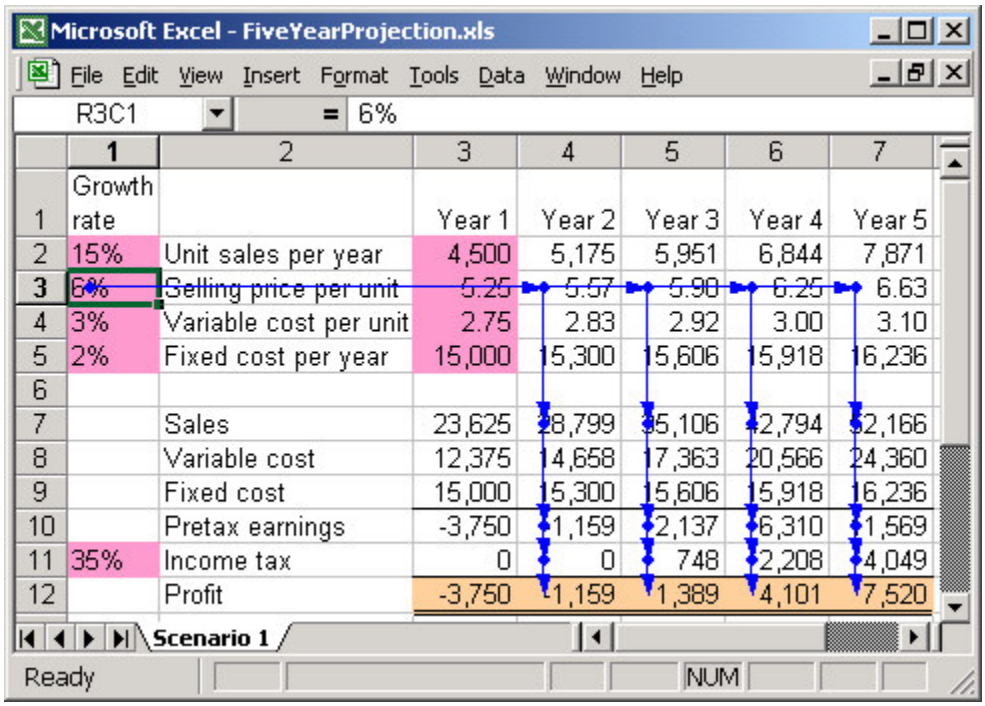


	1	2	3	4	5	6	7
1	Growth rate		Year 1	Year 2	Year 3	Year 4	Year 5
2	15%	Unit sales per year	4,500	5,175	5,951	6,844	7,871
3	6%	Selling price per unit	5.25	5.57	5.90	6.25	6.63
4	3%	Variable cost per unit	2.75	2.83	2.92	3.00	3.10
5	2%	Fixed cost per year	15,000	15,300	15,606	15,918	16,236
6							
7		Sales	23,625	28,799	35,106	42,794	52,166
8		Variable cost	12,375	14,658	17,363	20,566	24,360
9		Fixed cost	15,000	15,300	15,606	15,918	16,236
10		Pretax earnings	-3,750	-1,159	2,137	6,310	11,569
11	35%	Income tax	0	0	748	2,208	4,049
12		Profit	-3,750	-1,159	1,389	4,101	7,520

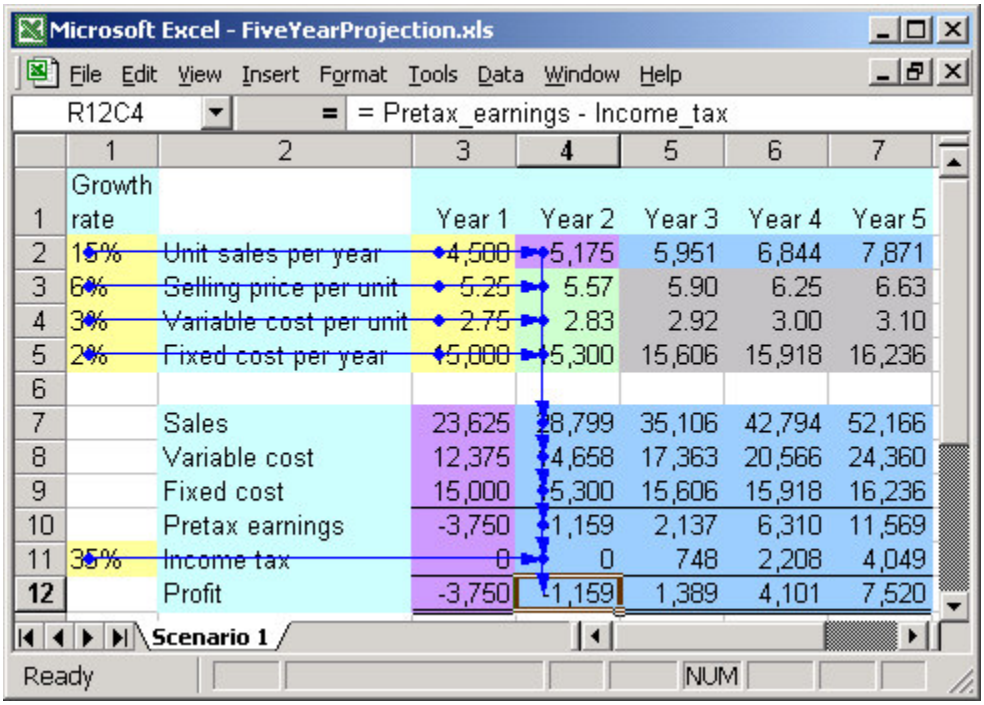
Sometimes it is important to see the precedent/dependent graph, superimposed visually atop the cells of the spreadsheet. Excel provides a tool to show precedents and dependents, but it is hidden on the auditing toolbar and is not particularly useful. The CellMaps toolbar provides an improved version of the standard Excel tool, one that shows all the precedents and dependents of a cell at the same time, all the way to the ends of the precedent/dependent graph. To use the CellMaps tool, select the desired cell or cells, then click the '+' arrow button; turn off all the arrows by clicking the '–' arrow button. One particularly clever use of this tool is to select all the output cells, then click the Show Arrows button to see the entire precedent/dependent graph for the entire spreadsheet; if the spreadsheet developer did a good job, you should see arrows that flow left-to-right and top-to-bottom with no crossings, though of course there are some circumstances where that goal is simply not possible:



Another common use of the Show Arrows button is to trace all the uses of a particular input cell. Here, cell R3C1, the growth rate for selling price per unit, is traced to the calculation of the selling price per unit for each of the four out-years of the forecast, and thence to the calculation of sales, pretax earnings, and profit:

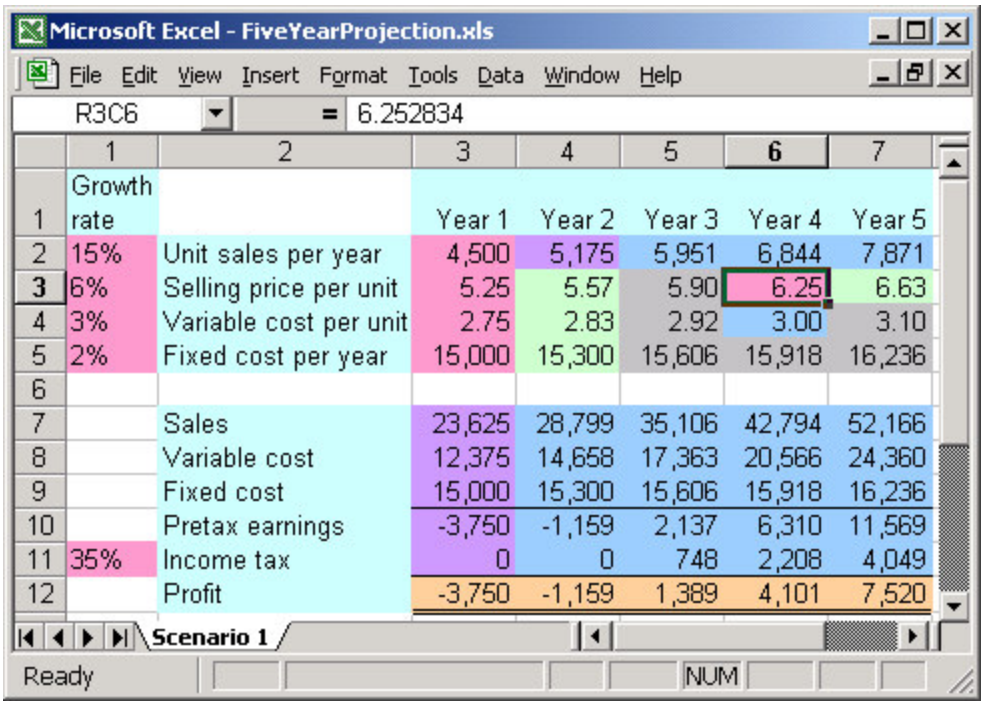


Here is one more view of the sample spreadsheet, with arrows added to show the precedents of the D12 output cell; note that profit depends on all nine of the input cells:



All of this coloring makes it obvious how to check this spreadsheet. Verify that the nine rose input cells (or yellow number cells — in this case they are exactly the same) contain the desired values. Verify that the seven lavender formula cells contain the desired formulas. Verify that the structure of the copied formulas (the blue, green and gray cells) makes sense. And you're done. That's a reduction from fifty-five cells in the precedent/dependent call-graph to only sixteen that need to be checked, because the structure of the spreadsheet shows copied formulas that don't need to be individually checked.

Although this discussion has shown how to use CellMaps to reduce the work of checking a spreadsheet, CellMaps can also be used to help you understand the structure of an unfamiliar spreadsheet. Here's a tip: the colors are perfectly visible even when the spreadsheet is zoomed to 10%, making it easy to see the structure of a large spreadsheet without a lot of panning around the screen. And here's another tip: a common error when developing spreadsheets is to inadvertently replace a formula with its value; that error has a very obvious signature when the spreadsheet is colored, as in the situation below, where cell R3C6 is the culprit:



	1	2	3	4	5	6	7
1	Growth rate		Year 1	Year 2	Year 3	Year 4	Year 5
2	15%	Unit sales per year	4,500	5,175	5,951	6,844	7,871
3	6%	Selling price per unit	5.25	5.57	5.90	6.25	6.63
4	3%	Variable cost per unit	2.75	2.83	2.92	3.00	3.10
5	2%	Fixed cost per year	15,000	15,300	15,606	15,918	16,236
6							
7		Sales	23,625	28,799	35,106	42,794	52,166
8		Variable cost	12,375	14,658	17,363	20,566	24,360
9		Fixed cost	15,000	15,300	15,606	15,918	16,236
10		Pretax earnings	-3,750	-1,159	2,137	6,310	11,569
11	35%	Income tax	0	0	748	2,208	4,049
12		Profit	-3,750	-1,159	1,389	4,101	7,520

TrafficLights

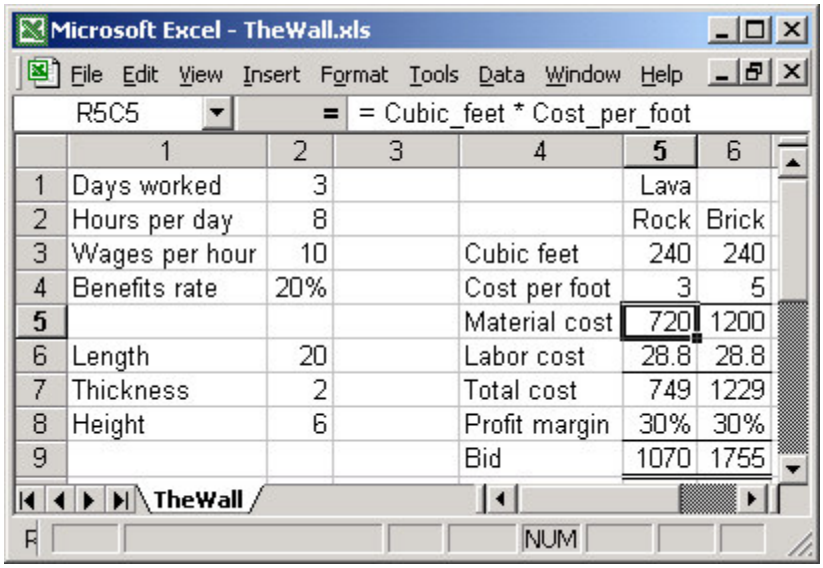
The normal antidote to spreadsheet errors is cell-by-cell inspection of a spreadsheet by an independent checker, someone other than the original spreadsheet author. That's hard work, made harder by the need to ensure that each and every cell gets checked. The problem is that the graph of cell dependencies doesn't necessarily flow top-to-bottom or left-to-right — the flow can meander like a stream, hither and yon, depending on the author's whims — making it hard to know what has already been checked and what is ready to be checked next.

The TrafficLights macros provide a systematic way to perform cell-by-cell spreadsheet inspection, relying on the visual cues of the three colors of a traffic light — green, yellow, and red — to guide the actions of the inspector. Green cells have been checked and are okay. Yellow cells are ready to be checked by virtue of the fact that all their predecessors are green. Red cells aren't ready to be checked because one or more of their predecessors aren't green. The inspector loads the spreadsheet to be checked, then prepares it for checking by running a macro that colors all cells with predecessors red and all cells with successors but no predecessors yellow, leaving all other cells uncolored. Then, the inspector repeatedly selects any yellow cell and checks it, fixing any errors and calling a second macro to color it green when it is okay. Besides changing the color from yellow to green, that second macro dynamically recomputes the colors of all the remaining cells in the spreadsheet, and the inspector chooses another yellow cell,

repeating the process until all cells are green, indicating the entire spreadsheet has been checked.

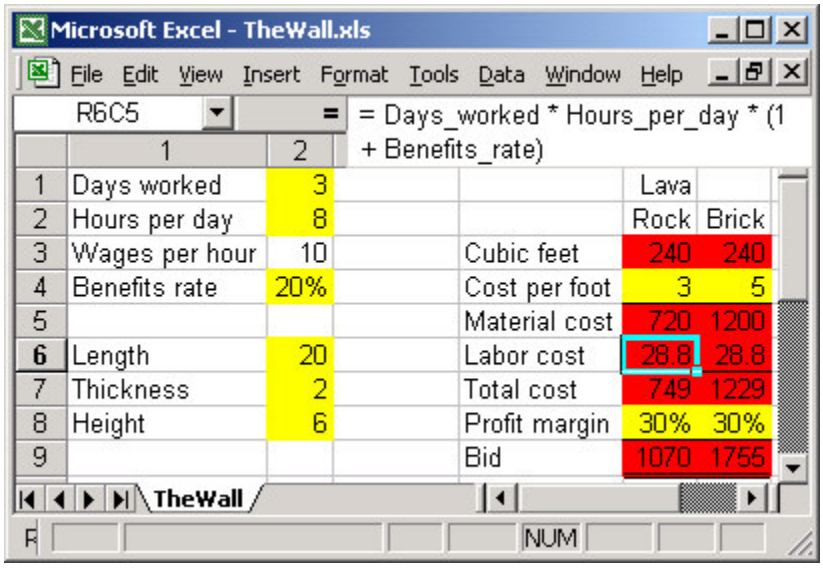
The TrafficLights toolbar has five icons, an "on" icon to set the initial colors, red, yellow and green icons that change the selected cells to the specified colors, and an "off" icon to reset all the cells to have a white background:

We'll use an example to illustrate use of the TrafficLights toolbar. The task is to build a spreadsheet model to help create a bid to build a wall. There are two options — lava rock or brick. Both walls will be built by crews of two, working three eight-hour days to build either type of wall. The wall will be 20 feet long, 6 feet tall, and 2 feet thick. Wages will be $10 per hour per person. You will have to add 20% to wages to cover fringe benefits. Lava rock will cost $3 per cubic foot. Brick will cost $2 per cubic foot. Your bid must add a profit margin of 30% to your expected cost. Consider the following spreadsheet, which has been seeded with errors:



	1	2	3	4	5	6
1	Days worked	3			Lava	
2	Hours per day	8			Rock	Brick
3	Wages per hour	10		Cubic feet	240	240
4	Benefits rate	20%		Cost per foot	3	5
5				Material cost	720	1200
6	Length	20		Labor cost	28.8	28.8
7	Thickness	2		Total cost	749	1229
8	Height	6		Profit margin	30%	30%
9				Bid	1070	1755

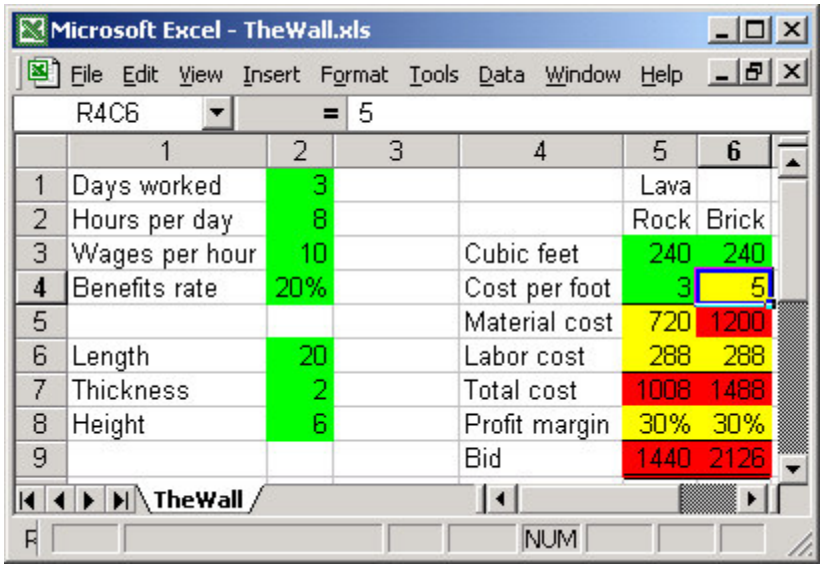
Start checking by clicking the "On" button on the TrafficLights toolbar. Input cells with no precedents are colored yellow, while formula cells that do have precedents are colored red, since none of the precedent cells have yet been checked. The spreadsheet looks like this:



	1	2			Lava Rock	Brick
1	Days worked	3				
2	Hours per day	8				
3	Wages per hour	10		Cubic feet	240	240
4	Benefits rate	20%		Cost per foot	3	5
5				Material cost	720	1200
6	Length	20		Labor cost	28.8	28.8
7	Thickness	2		Total cost	749	1229
8	Height	6		Profit margin	30%	30%
9				Bid	1070	1755

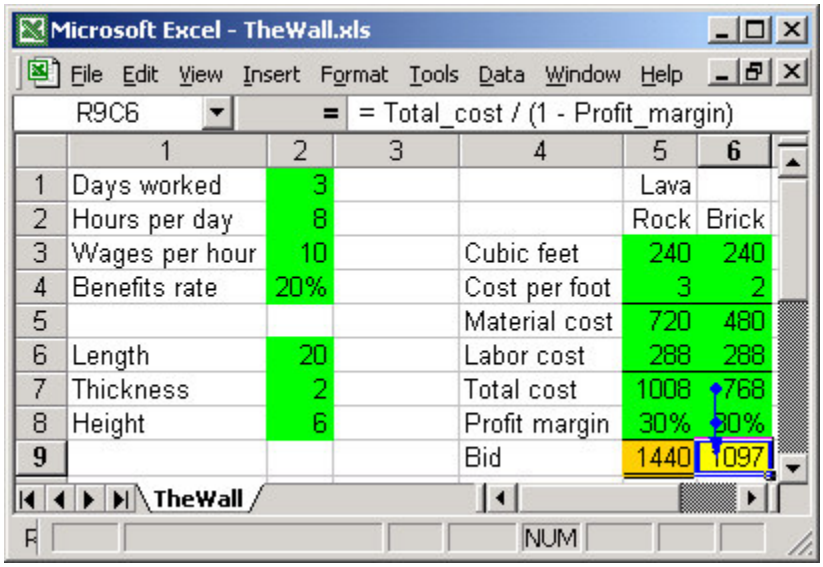
One error is immediately obvious. The wage rate cell in R4C2 is white, not yellow, indicating that it has no dependents. But clearly, labor cost must depend on the wage rate, and a glance at the formula in R6C5 confirms the omission. This error is easy to fix; just change the calculation of Labor_cost in cells R6C5:6 to read = Days_worked * Hours_per_day * Wage_rate * (1 + Fringe_benefits). Then select cell R4C2 and click the yellow traffic light to recolor the spreadsheet.

In the spreadsheet below, we have moved on in the checking process, completing all the checking of labor cost and materials cost except the cost per cubic foot of brick, which the project specifications state is $2, not $5. Since this cell is yellow, it has no precedents, and when we insert a constant in the cell it still has no precedents, so we can just make the change and click the green traffic light to color the cell green:



	1	2	3	4	5	6
1	Days worked	3			Lava	
2	Hours per day	8			Rock	Brick
3	Wages per hour	10		Cubic feet	240	240
4	Benefits rate	20%		Cost per foot	3	5
5				Material cost	720	1200
6	Length	20		Labor cost	288	288
7	Thickness	2		Total cost	1008	1488
8	Height	6		Profit margin	30%	30%
9				Bid	1440	2126

Now, we can proceed to check the rest of the cells. We fix the Bid formula in cells R9C5:6, which is = Total_cost / (1 – Profit_margin) but should be = Total_cost * (1 + Profit_margin) according to the specification.

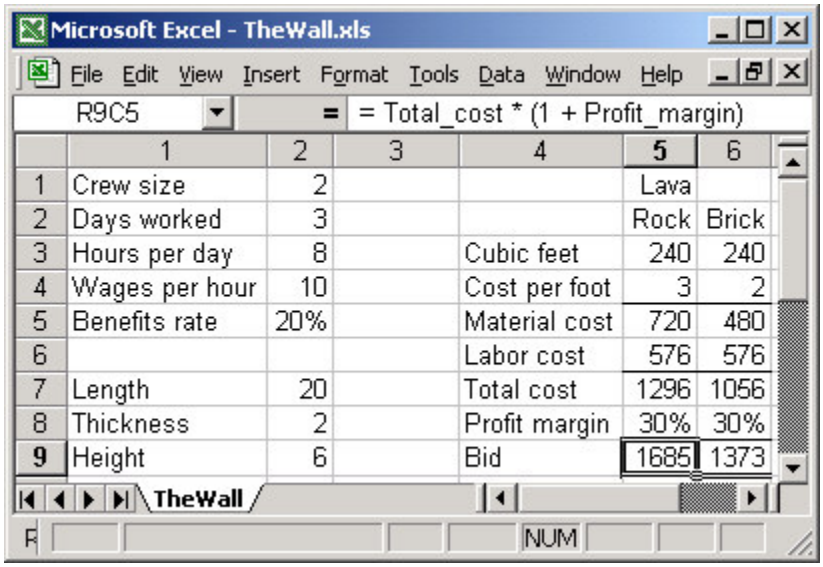


	1	2	3	4	5	6
1	Days worked	3			Lava	
2	Hours per day	8			Rock	Brick
3	Wages per hour	10		Cubic feet	240	240
4	Benefits rate	20%		Cost per foot	3	2
5				Material cost	720	480
6	Length	20		Labor cost	288	288
7	Thickness	2		Total cost	1008	768
8	Height	6		Profit margin	30%	30%
9				Bid	1440	1097

Although we haven't talked about it, if you have been following along with the example at your own computer you may have noticed that when you select a yellow cell the spreadsheet is decorated in two ways. First, precedent arrows for the selected cell appear, simplifying the task of determining what cells are used in the formula. Also, adjacent cells with identical formulas are colored a gold color, suggesting that the inspector might want to add those cells to the cell currently being inspected. Both these features appear in the spreadsheet shown above, where R9C6 is the selected cell, its precedents are R7C9 and R8C9, as indicated by

the two blue dots on the arrow, and adjacent cell R9C5 has the same formula and could be checked at the same time, as indicated by its gold interior.

Eventually, after checking all cells, the entire spreadsheet is green. Unfortunately, there is still an error. The original problem specification called for a crew size of two, so labor cost is half what it should be and the total bid price is incorrect. With crew size inserted as it belongs, the final, complete, correct spreadsheet is shown below:



	1	2	3	4	5	6
1	Crew size	2			Lava	
2	Days worked	3			Rock	Brick
3	Hours per day	8		Cubic feet	240	240
4	Wages per hour	10		Cost per foot	3	2
5	Benefits rate	20%		Material cost	720	480
6				Labor cost	576	576
7	Length	20		Total cost	1296	1056
8	Thickness	2		Profit margin	30%	30%
9	Height	6		Bid	1685	1373

The original spreadsheet contained four errors: wage rate omitted from the calculation of labor cost, wrong cost per cubic foot for brick, bid calculation incorrect, and missing crew size. The traffic lights guided us unerringly to the first three errors, but the fourth error, missing crew size, was found by comparing to the project specification, not by following the traffic lights. Omission errors like these are the hardest to find, with no cure other than diligent checking.

Sometimes the inspector will find and fix an error that causes the spreadsheet to change in such a way that the structure of the cell-dependency graph changes. This can happen, for instance, when a cell is deleted; in that case, you should color the cell green before deleting it so that its red successors will become yellow if they have no other predecessors. Another case arises when cells with successors but no predecessors are inserted; those cells should be colored with the yellow traffic light button so their successors can be properly recolored. The red traffic light button is used when the inspector changes the formula in a cell not knowing if its predecessors are green or yellow; it will properly color the cell either yellow or red, then recompute the colors of its successors.

Implementation Notes
For those who are curious, here is a brief discussion of the implementation of these two add-ins.

CellMaps is the simpler of the two, with a single code module. Input cells and output cells are determined by counting the number of cells in the C.Precedents and C.Dependents properties of each C in the ActiveSheet.UsedRange; cells with zero precedents and non-zero dependents are input cells, cells with zero dependents and non-zero precedents are output cells, and anything else is neither. The text, number, and formula buttons use the C.SpecialCells method of each C in the ActiveSheet.UsedRange to identify text, number, and formula cells; additionally, the C.FormulaR1C1 property of each formula cell is compared to adjacent cells, with equal C.FormulaR1C1 indicating a copy. The C.ColorIndex property of each C in ActiveSheet.UsedRange is set to zero to turn off all coloring. The draw-arrows button calls the C.ShowDependents method on each C in ActiveSheet.Selection, then recursively calls itself on each cell in the C.Dependents property of each of those cells, and likewise for C.ShowPrecedents and C.Precedents. C.ShowDependents and C.ShowPrecedents are also called to remove the arrows on request.

TrafficLights is more complicated, primarily because of the placement of precedent arrows and the gold color when yellow cells are selected, and their removal when the selection is moved. As with CellMaps, the Workbook_Open and Workbook_BeforeClose events are trapped in the ThisWorkbook module to create and destroy the TrafficLights toolbar, and code in the TrafficLights module uses the C.Dependents and C.Precedents properties of each C in the ActiveSheet.UsedRange to set and change colors as appropriate. The interest is in the TrafficLightsClass class module that handles the App_SheetSelectionChange event. The code itself is undistinguished. What is interesting is that a class module is needed because the SelectionChange event must be handled at the Application level; handling it at the Worksheet level doesn't work because the worksheet-level trigger calls code stored in the user workbook but handling the trigger at the application level calls code stored in the add-in, which of course is what the task requires.

Bibliography: The Wall task is taken from Ray Panko's two articles, which provide an excellent but scary introduction to the topic of spreadsheet errors: Raymond R. Panko. Hitting the Wall: Errors in Developing and Debugging a "Simple" Spreadsheet Model. *Proceedings of the 29th Annual Hawaii International Conference on System Sciences — 1996.* Raymond R. Panko and Ralph H. Sprague, Jr. Hitting the wall: errors in developing and code inspecting a 'simple' spreadsheet model. *Decision Support Systems* 22 (1998) 337–353. The trichromatic scheme for guiding spreadsheet checking is a simplification of a much more sophisticated inspection-guider described in Margaret Burnett and Gregg Rothermel's articles What You See Is What You Test and Spreadsheet Slicing: J. Reichwein, G. Rothermel, and M. Burnett, "Slicing Spreadsheets: An Integrated Methodology for Spreadsheet Testing and Debugging," *Conference on DomainSpecific Languages*, Austin, Texas, pp. 25-38, October 3-5, 1999. K. Rothermel, C. Cook, M. Burnett, J. Schonfeld, T. Green, and G. Rothermel, "WYSIWYT Testing in the Spreadsheet Paradigm: An Empirical Evaluation," TR 99-60-08, Oregon State University, Computer Science Department, December 1999. Advice on programming the macros came from John Walkenbach's well-written and very complete book about Excel/VBA programming: John Walkenbach. *Excel 2002 Power Programming with VBA*. John Wiley and Sons, 2001, ISBN 0764547992.