**Minefield - Task 4**

It is time to add the “game” elements into the application.

What actions and functionality are needed next?  
What sort of things make this into a game?

After a bit of thought, I decide that here is where we are at….

**Task still to be solved:**1. Create bomb sites  
– how will bomb locations be determined? (it needs an algorithm)  
– how will I store this information?  
– how many bombs will I set?  
– any other considerations?

**Task still to be solved:**2. Once the bomb locations are set, as the sprite is moved around we must…  
– add more functionality to count and display the number of adjacent bombs

**Task still to be solved:**3. When a move is requested…  
– we must determine of the sprite moved to a bombsite?  
– if true, decide what happens next?

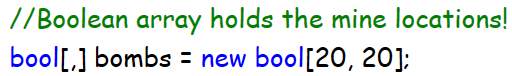
In the simple version supplied, the game ends and all bombsites are revealed  
So, how will we program this?

**Task still to be solved:**4. What happens if the sprite makes it to the target point?   
 (er, where is the target point for that matter?)

– show some sort of success message?  
– maybe provide the chance to start again?  
– maybe provide the chance to enter a fastest-times “hall of fame”?

**My Solutions:**

To store and track bombsites I used a 2D array of type bool. You could potentially use a list instead. There are undoubtedly several other ways to achieve this too.



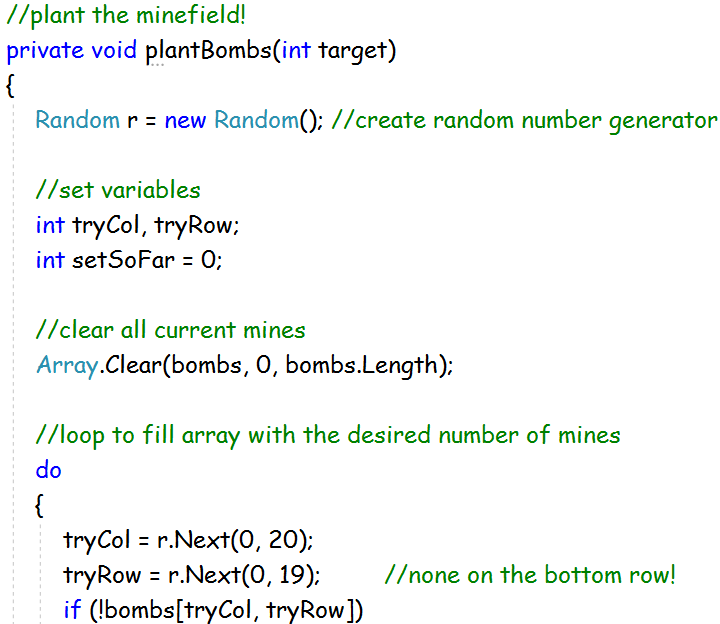
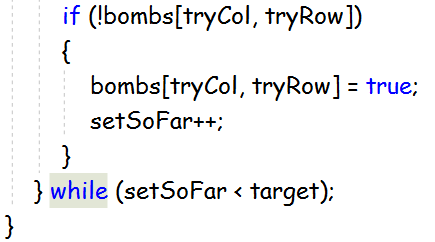
Next, I need to write a subroutine that sets the bombs when called. It will use the **Random** class to generate chance placements, as this is a key game element.

I need a loop of some sort to set *k* number of bombs. I don’t want any bombs to be stacked into the same grid cell, so it should only add to a cell if not already filled. Any other rules?

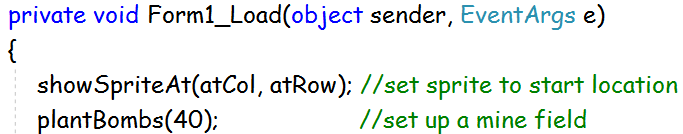
My code accepts an integer stating how many bombs to plant. This will make it easier to add levels later. More bombs make for a harder game.

It resets the array to empty. This makes it easier to program a reset/start button later.  
It uses a do-loop to select bombsites – just seemed like a chance to show off this loop style.

My code is provided below, but you are encouraged to try and develop your own first, and to only expand the images and use my code if you need too!

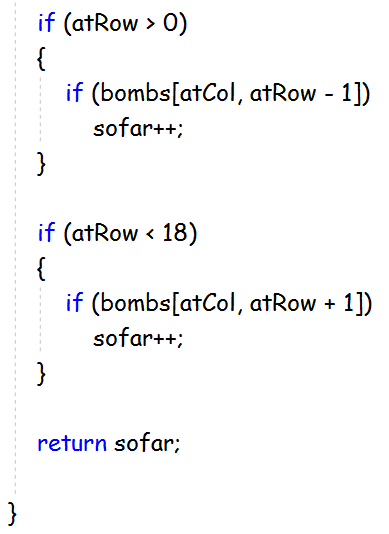
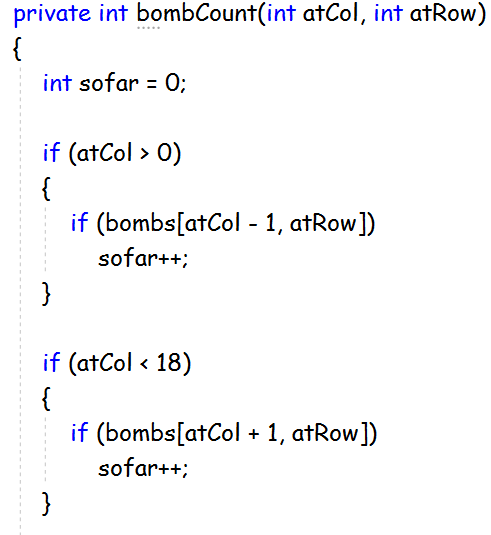
We should call this subroutine when the Form is first loaded…



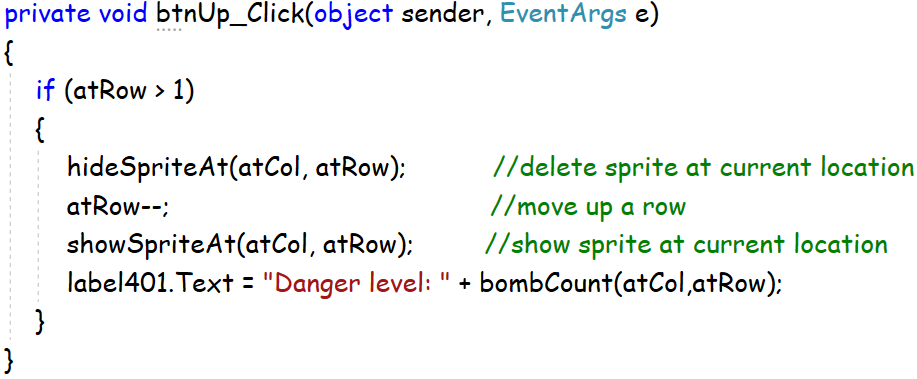
Next, we must add more functionality to the move sprite routines. The program should count and display the number of adjacent bombs whenever a move takes place. First, add a label to report how many bombs surround the current location recorded by (atCol, atRow)…



My code for counting bombs is presented below: as always, try to solve it yourself first.



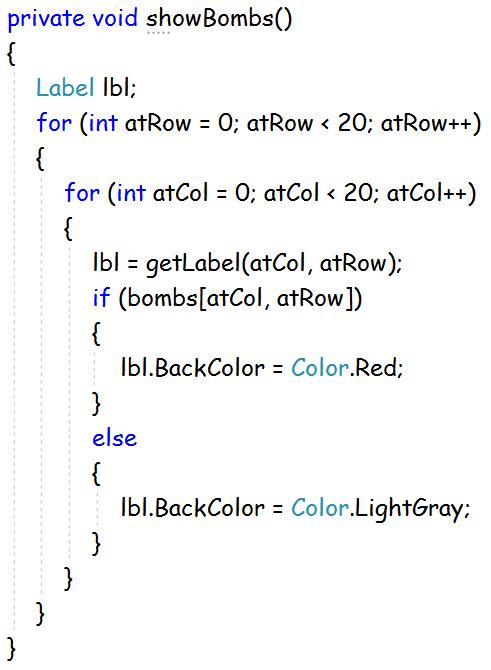
First, try calling this method / subroutine upon each move. For example…



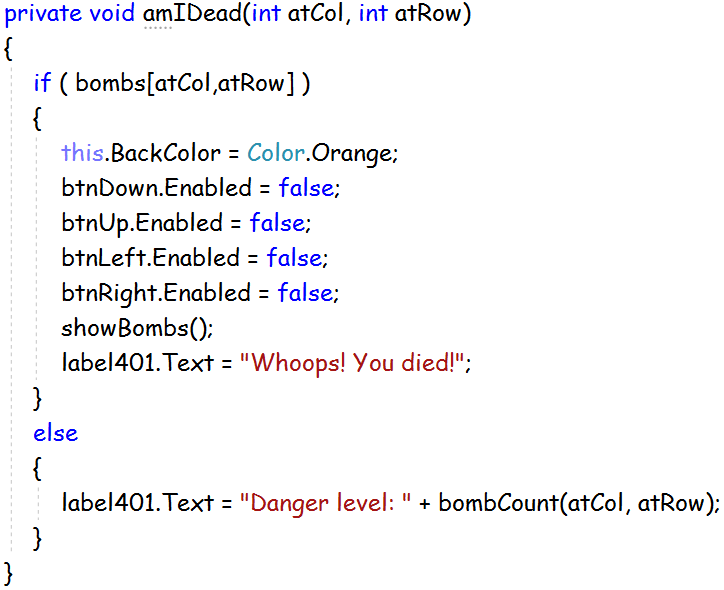
It will also need to be called after the bombs are first set.

Test the program. All well and good, but we do need to check if the move means that the sprite has landed on a bomb. If so, the game should end, and all the bombs revealed.

This requires a subroutine to displays all bombsites: see if you can work one out for yourself. If not, my code is provided below. It uses crude colour squares at this point, but a future improvement might be to make use of some nice images instead.

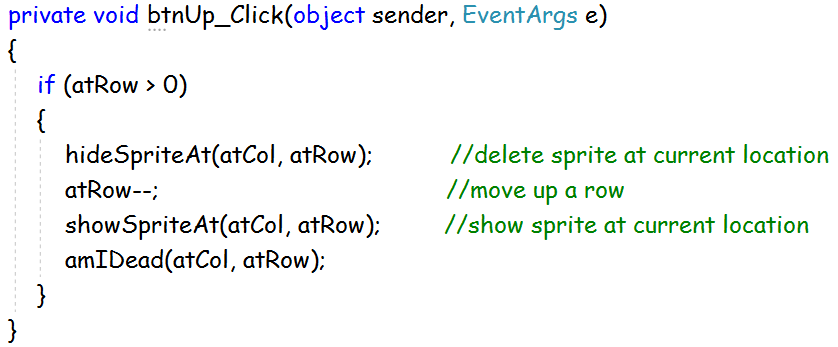


It also requires a subroutine (mine is called **amIdead**) to check if a move has resulted in hitting a bomb…



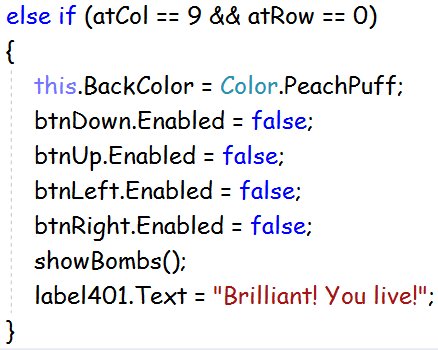
The final piece of coding is to sort out the sequence of operations when a move is requested.

Return to your **btnUp\_Click** event and modify it. First, it checks if the move is allowed (i.e. not already at the top of the grid). Then it hides sprite, updates the coordinates, and shows the sprite; and finally, it checks the move: if it was to a bombsite then it is game over, otherwise just update the adjacent bombs count label…



Add the same update to all the other movement buttons.

Currently, the game has no end-point. To solve this make a change to the **amIDead** subroutine to add a further check…



We are there! That’s all there is to coding up a basic minefield game!

<commit4>