Teaching Empirical Modelling Through Empirical Models

D.C. Gregson

February 1, 2011

Abstract

The radical shift from traditional programming to modelling can be conceptually dauting to computer scientists. This paper attempts to practically introduce the concepts of empirical modelling. It explores the principles behind this new perspective of computing. We develop a model of the well known logic game, Minesweeper, in an attempt to demonstrate the advantages of empirical modelling.

1 Introduction

For university students that have spent three years studying the principles of procedural and object-oriented programming, the task of having to approach the development of programs in a radically new way can be difficult. During my research into empirical modelling ("EM") and during the series of lectures I attended, I often found it helpful to mentally attach the new concepts and paradigms to a construal that I was familiar and comfortable with.

My mind often started attributing the concepts of EM to construct a mental model of the logic game: Minesweeper. Applying the philosophies to a mental model helped further understanding of EM. Once I understood the field, I realised that Minesweeper is an excellent example of a game that could benefit from taking an empirical modelling approach to its implementation.

This paper therefore attempts to construct a model of Minesweeper. We seek to extend the traditional game play to allow the user to explore the game state more fully, which demonstrates an advantage of the EM approach, whilst also allowing players to improve their puzzle solving abilities. We use Eden as the engine, as well DoNaLD and SCOUT for the GUI.

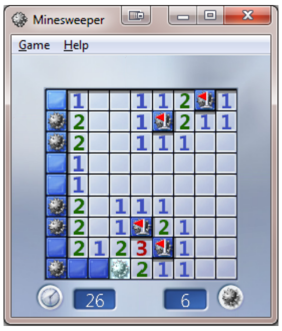

Figure 1: A construal of a minefield from Microsoft

2 Minesweeper

2.1 The Game

The game board is a minefield divided into small squares. Buried under some of these small squares are landmines. If the player explores this square, then the game is over, along with the player's life.

If the player explores other squares, one can use a mine detecting device to sense mines in close proximity. The detector can sense how many mines exist in the eight squares that surround the current square.

Logical inference can be used to deduce which squares must contain mines. The player plants a flag over the squares that are deduced to be mines, instead of revealing them. As the player is told how many mines exist on the minefield, once enough flags have been planted, the player is then invited to substantiate their skills in logic by exploring the rest of the minefield. If the player is correct, they should survive to tell the tale.

2.2 Suitability for an Empirical Model

Minesweeper is an excellent candidate for an empirical model. Even the existing forms of the game, that are implemented with traditional programming techniques, embody some of the principles of empirical modelling.

Inevitably and thankfully, a computer programmer cannot accurately simulate a real-world minefield. When designing a minesweeper game, a programmer must therefore create a more abstract construal that still represents the minefield, such that the player experiencing the game is still aware of the perils of exploring uncharted territory in a minefield.

Typically, the minefield construal is created by 'tiles' which are overturned. The tile artefacts model the areas on the minefield. The value of uncovered tiles are dependancies formed from the values of their neighbours on the grid. The player is invited to cautiously explore the model of the gamestate, and logically reason from the dependancies within. However, if the player makes false reasoning and finds a mine, the game is ended.

By extending the Minesweeper construal, we are able to abstract it away from the game entirely, and model the process of playing the game itself. We demonstrate to the player ways of thinking about how to solve the logic puzzle that is behind the minefield construal.

3 An Empirical Model of Minesweeper

As discussed, the existing forms of Minesweeper already embody some of the notions of empirical modelling. However, if we create a truly empirical model of the environment, rather than simply presenting a game, we are able to provide a model which allows the user to explore the consequences of making certain inferences from the state of the environment. It provides a more interactive way for the player to learn the dependancies involved in the logic that creates the puzzle.

In order to achieve this, we extend the interactivity of a traditional implementation by dynamically changing the values of our tile artefacts according to how many mines are both known and guessed to be in the vacinity of the square. In other words, the artefacts have dependancy on the observables created by the player, as well as the game state which is hidden to the player.

In this way, we create an open-ended exploratory environment for the player to explore. The player is free to move around the states of the model, and then submit a solution when satisfied. The model's submitted state can then be verified against the hidden game-state that is randomly created for each game.

Such a model allows the player to make 'mistakes' in their reasoning, and to learn from those mistakes. We highlight the mistakes by proving impossibility within the construal. For instance, if a square indicates that one unflagged mine still surrounds that square, and the player plants a flag in one that surrounds it, the square will decrement to indicate that there are no unflagged mines in the vacinity. If the player places another flag in the vacinity, then the counter would further decrement to -1. The player will intuitively recognise that this is nonsensical, as it is not possible to have -1 mines in the vacinity. The player than therefore infer that at least one of the mines surrounding that square must be incorrect.

If the player is skilled and makes no mistakes, then the model will also act as an aid for completing the game quickly. As the player moves game states by planting flags and exploring squares, the explored squares will decrement when required. Assuming all flags are correctly laid, any uncovered square displaying a 0 will allow the player to infer that they may explore any unexplored squares surrounding that one.

4 Evaluation

Modelling a game such as this proves useful, both as a learning tool for the logic puzzle and also as an introduction to empirical modelling. The model presents a more interactive experience than traditional implementations, and explains to the player how one makes inferrals to solve the puzzle.

5 Suggestions for Model Extensions

Grid size and mine densities Common implementations of Minesweeper allow for the player to adjust how large the grid is, or the level of peril the player is subject to.

Gameplay The technical challenges involved and time limitations lead to a rudimentary looking model. This model could be enhanced to give a better user experience. Some of these improvements will also improve the status of the model as a teaching method. For instance, many implementations of minesweeper use different colours for different squares, indicating which squares are in particularly perilous places. It is noted that the model we have implemented looks very little like a minefield, so enhancement of the model could help improve the chance of a succesful transition of the artefacts of the model into a construal of a minefield in the player's mind.

Puzzles Now that the model is created, it would be fairly simple to create similar puzzles with different rules. Perhaps our detector can find mines in the 24 squares surrounding our current location? Extending the rules like this is particularly interesting on an empirically modelled implementation, as we can fully explore the consequences of our changes.

6 Personal Observations

SCOUT

One of the limitations I found in creating a model like a game board is that there seems to be no inherent way of creating multiple SCOUT windows from similar code. This is due to the fact that there are no code loops supported within SCOUT. It was frustrating, and initially I had to write out 81 SCOUT definitions in order to create my grid. Not only was this tedious, but it meant that I did not have the ability to have a dynamic game grid size.

I am therefore thankful to an anonymous contributor to WEB-EM-2 [1]. This paper included a Su Doku game model. Some of the challenges that

[1] 0208873, Is Empirical Modelling Puzzling?

the Minesweeper model faced were also faced by the Su Doku model, and inspection of this model presented an interesting solution to being able to efficiently build a grid of SCOUT windows. It involves being able to embed SCOUT notation in the same script as EDEN notation. By working in EDEN, we are able to use EDEN's `execute();` function to run a script from a string. We can therefore use EDEN and its code loops to build a long string of repeating definitions by 'escaping' out of the string and then concatenating the results. Finally we run the string in an `execute();` statement. By using an intelligent solution like this, we are able to create a grid size of the player's choice, by offering an input to determine grid dimensions.

Although this was an effective solution to my problem, the solution itself could be considered a 'hack,' and it would be preferable to build functionality into SCOUT that facilitates simple creation of multiple windows.

Spreadsheets

My research into EM found an area within it that is particularly relevant to a model such as mine. The argument has been made that computer spreadsheets can be strongly likened to empirical models. [2] [3] [4] This argument is supported by a model like mine, which could be implemented on a spreadsheet, and indeed already has. [5] The research noted however that the differences between empirical models and spreadsheets include the fact that spreadsheets have 'sophisticated forms of dependency based on geometric organisation that are not easy to express in definitive scripts.' This conclusion is certainly one which this paper concurs with, as modelling the geometric organisation of the minefield construal proved difficult.

The ability to create artefact organisaton is one that I consider worth exploring further, as it could prove to be a powerful addition to notation language. However, it is perhaps best implemented as an additional notation like DoNaLD or SCOUT, rather than built inherently into EDEN. To do so in EDEN would reduce the flexibility that makes EDEN such an excellent tool to create empirical models.

7 Playing the Game

To explore the model, clicking on squares will detect mines in the vacinity. One can hover over a square and plant a flag by pressing 'f'. The observable game state dependancies will dynamically change when the flag is planted. One can remove a planted flag by pressing the 'space bar'.

[2]Beynon 1997, Empirical Modelling for Educational Technology

[3]Russ 1997, Empirical Modelling: The Computer as a Medium

[4]Harfield 2007, Empirical Modelling as a new paradigm for educational technology

[5]`http://lazyslug.homeip.net/minesxl.php`