Towers of Hanoi

0714405

Abstract

In education there has been a marked emphasis shift from problem solving via teaching through audio and visual methods to teaching through problem solving, or constructionist learning. In this paper we discuss the advantages and disadvantages of using Empirical Modelling as a tool to promote this form of learning and compare it to traditional IT solutions. As the core of this investigation the Towers of Hanoi was modelled using the Empirical Modelling environment EDEN. The basic puzzle was first implemented and then, exploiting the open-ended qualities inherent in Empirical Modelling, the model was extended. Although EM techniques are discovered to be useful with a large potential, the current implementations are not stable and functional enough for wider use.

1 Introduction

The idea of learning through interaction and experimentation is not a new idea, in fact, as far back as 450BC Chinese philosopher, Confucius is reported to have said: "tell me and I will forget, show me and I may remember, involve me and I will understand". But it is an idea that how gained greater and greater support recently. It is an easy step to see that this form of learning is ideally suited to computer technology, but despite this, technology has seen only a limited success in any sort of learning. Meurig Beynon argues that Empirical Modelling will change that (Beynon, 2007) due to the open nature of the models created and the close link between the development process and the learning process.

The Towers of Hanoi is a classic mathematical game with a basic rule set, but by modelling it using Empirical Modeling techniques we can extend the model dynamically, without a new game instance. This is done by building on the base dependancies set up when creating the model originally. These extensions are not only useful from a learning point of view but if successful should support EM claims.

The puzzle can be extended by varying a variety of variables and by relaxing some of the constraints of the base puzzle; such as the number of blocks, the number of bars and the widths of the blocks. We will examine if the development stage of EM is stable and 'friendly' enough to be used by developers and students alike.

2 Research

2.1 Constructionist Learning

Constructionist Learning, in essence 'learning by doing', is the latest catch phrase in educational circles, but the core ideas have been around for generations, as we saw by the quote from Confucius (Hein, 1991). There is a new acceptance of this set of ideas and to go with this new research to support it.

The idea is that learners, through experimentation and experience, can construct knowledge for themselves (Brooks, 1993). Technology is an obvious choice for implementing this type of learning.

2.2 Empirical Modelling

The standard use of technology in learning has been through the development of a model which a teacher runs and then is given to a student to use. If the teacher wants to change an aspect of the model it is either very difficult, requiring knowledge of traditional programming, or impossible and the model needs to be sent back to the developer. Because of this, Educational Software has had very limited success (H Jonassen, 1999).

One of Empirical Modelling's aims is to bridge the gap between the creation and programming of a model and use of it by making the development part of the learning process, applying the fundamental idea behind constructionist learning. By creating the model the user will learn about it. (Beynon W. M.)

While the same could be said for traditional programming there is actually a big gap between the idea behind the model and the way you would go about programming it. EM modelling on the other hand provides a continuous connection with personal experience.

3 Towers of Hanoi Model

The Towers of Hanoi puzzle consists of three bars, or towers, and a number of blocks of different sizes. The blocks are initially arranged in ascending order of size on bar 1 so that the smallest block is on top and the largest is at the bottom. The aim is to move the entire stack to another bar.

To achieve this goal the player can choose to move the top block from one bar to another, as long as this move doesn't result in a larger block being placed on a smaller.

The optimal solution to the Towers of Hanoi game, ie the minimun number of moves needed given x blocks to reach the solution, is well known and proved: 2^x-1. But by loosing certain constraints, such as the number of bars, the problem can be made significantly more difficult. For instance the solution to the Towers of Hanoi with four bars (called Reve's Puzzle) still an open problem.

Through implementing this puzzle and the planned extensions it is possible for a teacher to teach the student the mathematical skills needed to complete the puzzle, such as solving recursion and basic logic. By implementing and working on the extensions the student can then gain experience through empirical applications of these skills.

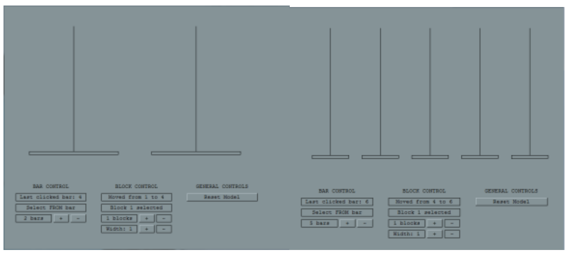
Following is a high level description of how the model was created. It focuses more on techniques used rather than being a detailed implementation walkthrough. Please refer to the source code to see the exact implementation of the following techniques.

3.1 Basic Dependencies and Procedures

The model is built primarily using EDEN and SCOUT, with the bars themselves as DONALD windows, to allow for easily scalable bars. There are originally 6 bars and 10 blocks created as SCOUT windows.

All the bars and blocks exist but only as many as are declared to exist are visible. So the first step was to tie the blocks and bars visibility to the EDEN variables 'noBlocks' and 'noBars'. This is done by tying their height and x coordinate to whether the item should be visible or not. Booleans are represented in EDEN by either a '1' for true or '0' for false. So by multiplying this value by the height of an item we are able to get a situation where they have a height of 0 if the item shouldn't be present.

The x-coordinates of the bars also had to be tied to 'noBars'. This is so that no matter what the value of 'noBars' the visible bars always take up the same space. See below for an example. These x-coordinate were then tied to the total desired width of the bars, so that if the user wanted to change the total width it would simply be a case of changing the 'totalWidth' variable.



The x-coordinates of the blocks also had to be tied to the position of the bar that they were stacked on. This was done by creating a list which stored the attributes of each block, called 'blocks'. One of these attributes was the bar that the block was currently stacked on. This was then used along with the blocks width, another one of the attributes stored, and the bars width, which itself was the result of a function on 'totalWidth'.

The block's y-coordinate is tied to a function on the block's relative height, the relative height itself was stored in the list mentioned above. The relative height was given as 1 for the bottom block on a bar, 2 for the next one up etc. The block's height is changed by one of procedures that run when the users make a change to the model.

The only procedure used by the basic model was to move a block from one bar to another. The user can do this by clicking on one bar, which contains the top block that they want to move, and then clicking on another bar, to which they want to move the block.

The procedure called, 'moveBlock', first checks if the move is allowed; if the block being moved is less than or equal to the top block of the new bar. The top block of a bar is stored in a list of bars. Once it determines the move is allowed, the top block of both bars is updated in the list 'bars', and the block's current block and height are updated in the list 'blocks'. Due to the already defined dependencies this moves the block.

The basic puzzle is set up using the procedure 'reset', which creates initial values for the blocks' height and width. It sets noBlocks and noBars to 3, and puts all blocks on bar 1.

3.2 Implemented Extensions

The dependencies listed above creates a construal of the basic Towers of Hanoi problem, but due to the nature of empirical modelling it can be extended with relative ease.

The easiest of these extensions was to allow the user to add and remove bars, to a minimum of 1 and a maximum of 6. This was done by creating a simple button that incremented the 'noBars' variable and another one that decremented it. Because of the already added dependencies this will move the blocks and bars to their relative positions regardless of which bar the block is on.

The other two extensions that were to be added were the ability to change the width, size, and to change the total number of blocks in the puzzle. Because they needed to happen dynamically, in other words without resetting the model, they were quite challenging to implement. Because of this certain assumptions where made.

The main assumption was that at no time, even when changingthe width of a block or by adding a new block, the rule that a larger block can't be on top of a smaller block of the model should not be broken. A new block is always added to the bottom of bar 1. But as the blocks already in the tower have dynamic widths, this could lead to a situation where the new block could be smaller than another block in the stack. To get round this, when adding a block all the blocks above it are compared to its width and their relative heights are affected depending on the result.

Removing a block is much easier, simply the block with the highest index is removed and the relative height of other blocks in its stack are affected. The adding and removing of blocks is done by clicking an increment or decrement button in the control panel.

Decreasing the width of a block is very similar to adding a new block, in fact it is slightly simpler as the only thing that needs to happen is if the block above the changed block in the stack is now larger, then their heights should be switched. Because of the dependancies already described, changing the width of a block automatically re-centres it on the bar.

Increasing the width is the same except that it compares its new width to the width of the block below it in the stack. The user select which block they want to edit by clicking on it, they can then increase or decrease its width by clicking on increment or decrement buttons. The default block selected is the first one.

3.3 Difficulties Encountered

3.3.1 EDEN

Eden is very much a work in progress and as with any new environment which is still in its infancy it is limited in its implemented functionality. The model had originally been planned to allow for 'x' bars and 'y' blocks where 'x' and 'y' could be any positive integer. But for this to be allowed the windows in SCOUT would have had to be defined dynamically with a dynamically declared names.

If it were possible to use arrays, or some other list data type in SCOUT, then it would have been a simple case of extending the model to allow this. Especially considering that the block relative height dependencies were setup to allow for any number of blocks and bars, and for that matter any size width.

As a way round this problem a limit was set on the number of blocks and bars, so that each one could be hard coded. A solution which although necessary was undesirable.

Another improvement that would have been helpful, although not neccesary, could be the added functionality provided by allowing basic object orientation. Many of the defined variables and windows had identical structure with one or two variables changed.

3.3.2 Towers of Hanoi

While the basic dependencies from the Tower of Hanoi were implemented without too much difficulty. Creating the dependencies which kept the block heights correct proved to be much more difficult, especially when changing the number of blocks and the width of the blocks. The problem was solved by creating a complex interweaving of lists which monitor the block heights and widths.

3.4 Future Extensions

Because of the nature of EM, extending the model further isn't too difficult as all the old dependencies still apply. One possible extension could be to loosen, or tighten, the basic restrictions of the game, so to allow the user to choose the rules regarding if larger blocks can be stacked on top of smaller blocks. For example, the user could state that a block can only be stacked on a smaller one if it is only 1 unit bigger, or 2 units bigger.

As mentioned above the main extension that would improve the model the most would be to allow for 'n' blocks and 'n' bars. The dependencies to allow this are already in place. The problem is that EDEN does not yet have the required functionality to make this feasible.

4 Evaluation

The final model can be considered a success, with all the dependencies working well together, as would be expected, and all the planned functionality works, on top of that there is still room for extension and the techniques demonstrated could be transferred to the implementation of a different model.

4.1 Evaluation of EM

Although the model reached all of its aims, the main reason for its implementation was to determine the usefulness of EM as a way of creating educational models. We will now evaluate it from the point of view of the three potential users and compare it to traditional software development tools.

4.1.1 Developer's View

From a developer's point of view EDEN is a very good tool. For creating a model which is dependency based it proved to be very successful. Although a steep learning curve, made more difficult due to the poor documentation, once learned models of these types proved to be much easier to create than using standard programming tools.

This was helped even more by the close tie between the construction of a model using empirical modelling and the logical setup of the puzzle. In other words from a developer's point of view the way the model is created and used are very similar, much more so than with traditional tools. So if the modeller also wished to use the model he was creating then EM would be a good choice.

4.1.2 Teacher's View

From a teacher's view setting up the model would be very simple provided the developer had set up the dependencies correctly and created an easy to use user interface.

But if the teacher wanted to edit the model in a way not easily provided by the developer through the user interface they might find it difficult. Simple steps like changing the value of one variable wouldn't be too difficult but changing the model in a more significant way would be too much of a learning curve.

In fact from the teacher's point of view a well-made model using a programming tool could be just as good as a model made using empirical modelling. Albeit if the user was proficient in both traditional programming techniques and in EM techniques it would be easier to use EM.

4.1.3 Learner's View

If the three groups are kept distinct, teacher, developer and student there is no advantage to the student to use an EM model over another model.

But the greatest advantage of using EM is that these three roles could potentially be combined. Although it is quite a steep learning curve to learn how to program using EDEN or another EM environment, once the user has done so developing teaching and learning could all be combined.

As there is such a close link between developing using EM and the constructionist learning philosophy, the simple process of creating the model would be synonymous with the learning.

5 Conclusion

Throughout this paper we have explored the ideas surrounding creating educational models, in respect to constructionist learning. We have looked at the claims of EM and developed and evaluated a model of the Towers of Hanoi using the EM environment EDEN.

Empirical Modelling has a lot of potential. In fact the fundamental idea behind it has a lot of validity, especially when applied to learning. It is evident that there are close ties between empirical learning and model development using EM tools, to the point where the open- ended creation of models can be considered learning. This is a great advantage over traditional software development tools.

Through the creation of a model in EDEN of the Towers of Hanoi we have seen how the understanding of a problem can lead to the creation of a model, but more importantly that the flip side is also true. The creation of a model and the way new dependencies are directly affected by old dependencies can lead to a greater understanding of the problem that is being modelled.

But there is still a long way to go before these EM techniques can in traditional learning environments, such as schools. The current environments that use EM techniques, in particular EDEN which we have looked at in this paper, are too complex and not stable enough for the average learner, or teacher, to use. It will be interesting to see how these tools develop over the next few years.

References

Beynon, M. (2007). Computing technology for learning - in need of a radical new conception . *Educational Technology & Society* .

Beynon, W. M. (n.d.). Empirical Modelling for Educational Technology. *Department of Computer Science, University of Warwick* .

Brooks, J. B. (1993). *In search of understanding: the case for constructivist classrooms.* Alexandria: Association for Supervision and Curriculum Development.

H Jonassen, D. P. (1999). Learning with technology: A constructivist perspective. *Special Education Technology* .
Hein, G. (1991). *Constructivist Learning Theory.*