Learning Computer Science using Games and Puzzles

Paul Curzon
Middlesex University
p.curzon@mdx.ac.uk

Abstract

Many children's games have similarities to the structures we teach in Computer Science and those structures are chosen for similar reasons. Strategies for winning some games share properties with efficient algorithms. General lessons, such as why finiteness and determinism are important properties of algorithms, can also be found in games. We explore the idea of using games to teach introductory computer science and describe initial evidence evaluating the approach.

Many children's games have similarities to the structures we teach in Computer Science and those structures are chosen for similar reasons. For example, standard race game boards are lists - processed from start to end. More interesting games use more interesting structures. A circular list is found in Monopoly: the game could never end. Snakes and Ladders uses a directed graph. A treasure hunt is a traversal of a linked list. Stacks are so important that they abound in childhood, from the toys consisting of poles and rings we give to toddlers to the Tower of Hanoi puzzle. The similarities are not surprising since abstract data types model structures from the real world, as do games.

General lessons about algorithms can also be found in games. For example, the aim of Patience is to sort a pack of cards. Are its rules an algorithm? It illustrates why finiteness and determinism are important properties of algorithms. The importance of choice of representation can be demonstrated by, for example, the games of Spit-Not-So and Nim. In Spit-Not-So nine cards are placed face up. Each has on it one of the words: Spit, Not, So, Fat, Fop, As, If, In, Pan. Players take turns to pick a card. The aim is to be the first player to collect all cards containing a particular letter. A game might thus go:

PLAYER 1: SPIT
PLAYER 2: SO
PLAYER 1: FAT
PLAYER 2: NOT
PLAYER 1: FOP
PLAYER 2: IF
PLAYER 1: PAN

At this point Player 1 wins as they have the cards: SPIT, FOP and PAN – all the Ps. This game is equivalent to Noughts and Crosses/Tic-Tac-Toe [2]. Changing the representation to a 3-by-3 grid with a word in each cell makes the game suddenly easier. For example, the above game will end with your table looking like the following:

NOT **O**	IN	PAN **X**
SO **O**	SPIT **X**	AS
FOP **X**	IF **O**	FAT **X**

Nim consists of three piles of matches. Players take turns to remove any number of matches from one pile. The winner is the player who takes the last match. Winning moves can most easily be identified if the piles are represented using binary numbers. Winning moves are ones where the addition-without-carry of the three numbers of the resulting position is zero. Choose a good representation and you win the game.

20-Questions illustrates why binary search is faster than linear search. Would you start by asking "Is it Michelle Pfeiffer?" or would you ask questions such as "Male or Female?" that halve the number of people left whatever the answer? The most successful players are the ones who come up with a series of questions that approximate a binary search.

We can conversely design new games by starting from Computer Science. For example, let us invent a game based on Heaps. In Patience, the seven stacks of cards are arranged as an array. Cards can be moved between any of the stacks. In our newly invented “Heap Patience” the stacks are arranged as a binary tree. Cards can only be moved to the top of their parent’s stack. In addition, the face up part of any stack can be exchanged with its parent, provided the top card is greater than the top card on the parent stack. The stacks thus act together like a heap with high cards moving to the root of the heap. Playing it provides the basis for an understanding of Heaps. Rather than teaching it to undergraduates, teach it to children.

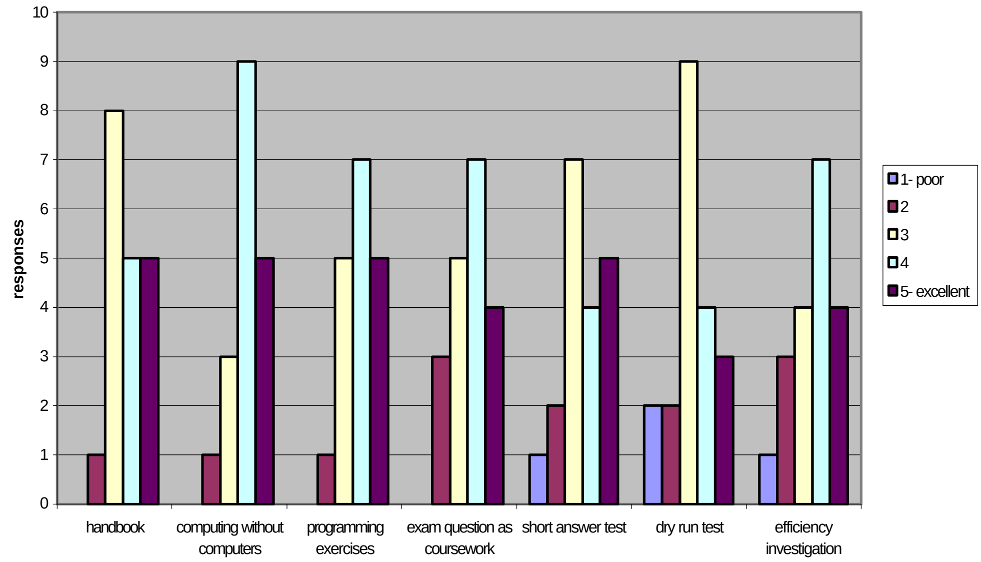
This idea of using games to teach introductory computer science have been piloted at Middlesex University on first year programming (approximately 250 students) and second year data structures and algorithms (50 students) modules. Examples based on games have been used in lectures to explain ideas before looking at them in computer terms. A booklet [3] containing a wide range of examples related to data structures and algorithms, based both on games and other non-computer situations was made available to students in addition to the normal module handbook. Also a lecture on binary search taught using 20-questions has been given on several occasions in 2000 at Open Days for School students (approximately 300-400 students in total) who had applied to Middlesex to do Computing courses.

Anecdotal evidence suggests that the approach works. Student results on the second year module have been high, and for the first years improving as more of this kind of example was introduced. The student feedback has been overwhelmingly positive with respect to lecture delivery. The first and second year students gave on average satisfaction ratings of 4 out of 5, for example. Written comments from the First Years suggested that the use of examples form everyday life were very useful in understanding computing concepts:

> "...metaphors and examples are excellent"
>
> "[the most useful part of the module was that] the lecturer was able to show demonstrations using everyday examples."
>
> "lecture well-constructed using appropriate examples of everyday life"

This approach did not suit everyone however: one student singled out the examples as being "useless and meaningless". A separate survey compared the booklet on data structures and algorithms in every day life [3], the module handbook and various assessed exercises given to the second years. It asked which of these the students found most useful in helping them meet the learning objectives of the module. They gave the highest rating to the booklet (see Figure 1). Open day feedback was similarly positive. One mother who accompanied her son to the lecture had used computers at work but had expected to find the lecture boring as she had little interest in computers. However, afterwards she was very positive about the lecture and was glad her son had made her stay. This feedback suggests that use of examples from everyday life (as opposed to specifically games) is an effective teaching tool. Games do however provide a rich seam of such examples. One reason the approach may be effective is that it provides opportunity for interaction into lectures. To be successful, however, it is important that it is made clear how the examples relate to the learning objectives of the module.

Figure 1: Which resource/assessment was most useful for meeting the learning objectives



Childhood is an excellent training ground for computer scientists. By this we do not mean that good games players will make the best computer scientists. Rather we suggest that the world of games and puzzles is full of hooks upon which the learning of computer science can be hung. Bell *et al*. [1] demonstrated a similar idea, developing activities for children that teach computing without using computers. We suggest that existing games use the same underlying structures as the data structures of Computer Science, their aim is often similar to the aim of common algorithms, and in some cases the best play is that which most successfully approximates the best algorithms. The more games and puzzles a person knows, the greater the foundation upon which the teaching of data structures and algorithms can be built. Games developed from Computer Science can both be fun and provide the foundations for learning the subject. We have looked at links between games and data structures and algorithms. It may also be possible to identify or design games with links to other aspects of Computer Science. We are currently using games to teach data structures and algorithms. With a longer-term view we should be designing new games that have deeper relationships with Computer Science concepts. We should be teaching them to children to provide the basis for them to learn Computer Science in the future.

References

1. T. Bell, I. Witten and M. Fellows, *Computer Science Unplugged*, http://unplugged.canterbury.ac.nz
2. E.R. Berlekamp, J.H. Conway and R.K. Guy, *Winning Ways*, V.2, Ch.22, Academic Press, 1982.
3. P. Curzon, *Computing Without Computers in Everyday Life*, Version 0.2, January 2000.

An earlier version of this paper was presented at the Computers and Fun II Workshop, York, December 1999.