1-1-2006

Reusable and Sharable Learning Objects Supporting Students' Learning of Data Structures in University Courses

K Chansilp
Suranaree University of Technology

Ron Oliver
Edith Cowan University

Chanslip, K. and Oliver, R. Suranee University of Technology, Thailand and Edith Cowan University, Australia. Reusable and Sharable Learning Objects Supporting Students' Learning of Data Structures in University Courses

K Chansilp[1] and R Oliver[2]

[1]School of Computer Engineering
Suranaree University of Technology, Thailand,
E-mail: kacha@sut.ac.th

[2]School of Communications & Contemporary Arts
Edith Cowan University, Australia,
E-mail: r.oliver@ecu.edu.au

ABSTRACT

Data structures are a conceptually demanding topic which confronts many computer science students early in their course. The topic has a strong conceptual basis and often proves difficult for many to grasp. This paper reports on a project which has developed a range of learning objects to help students learn about the different data structures and the algorithms by which they are controlled. Called VIDSAA, the suite of learning objects provides a visual representation which enables students to observe and interact with a large number of data structure algorithms as they are run and to observe and view the outcomes. The objects have been designed to enable students to explore and investigate the data structures as a means of developing their knowledge and understanding. The paper describes the design and development strategies that underpinned the development of the learning objects and showcases the resulting products. It discusses a project to explore how teachers and students might use the objects and the support they provide for learning.

INTRODUCTION

Data structures is a conceptually demanding topic which confronts many computer science and engineering students early in their courses. Students can have difficulty in understanding some programming concepts. They need to visualise what is actually happening inside of the computer memory when each statement of the program is executed. The problem is also caused by the change processes inside the computer. Students can often not fully understand these because they cannot see what is going on inside the computer (du Boulay, 1986, Mulholland and Eisenstadt, 1998).

Contemporary technologies appear to provide many opportunities to seek to address the learning needs of novice programmers in relation to learning difficult computer programming concepts. Kann, Lindeman & Heller (1997) suggest that the practice of using graphic representation of algorithms in textbooks provides constructs that are too abstract to aid learners to develop the logical thinking required in computer science courses. Many students who finish introductory classes, are still weak in their understanding of basic concepts. Students typically differ in their ability to comprehend and understand abstract material due to their inability to visualize the concepts. Previous research has proposed a number of ways to improve instructional materials and therefore student outcomes. For example, instruction can be made to incorporate dynamic explanations and conceptual maps can be developed to help students visualise the steps in program execution (Karsten & Kaparthi, 1998; Lischner, 2000; Rowe & Thorburn, 1999).

COMPUTER-BASED INTERACTIONS

Recent research into the cognitive load associated with student learning suggests a number of advantages can be gained in learning computer programming through the informed design of computer-based visualisations (Sweller, 1998). This research has demonstrated that when students are exposed to difficult concepts, their understanding can be enhanced through representations that enable the elements to be connected and linked by learners in ways that limit the extent of the cognitive load. When students learn from diagrams in textbooks, often their attention is split between explanations and the visual elements. The split-attention effect (Mayer & Anderson, 1991) has been shown to hinder learning but can be reduced through the use of imagery that combines the appropriate instructional text within the diagrams (Chandler & Sweller, 1992). In associated research, it has also been found that detailed textual descriptions accompanying graphics and images creates redundant information which also can impose extra load on students' learning (Chandler & Sweller, 1991). These difficulties can be overcome through representations which reduce the information redundancy, for example, through interactive applications which reduce the need for descriptions.

This provides an opportunity to investigate ways to enhance learning through the informed use of contemporary graphics programs. With emerging technologies, there are many ways to improve instructional materials and to help instructors and students improve the teaching and learning environment. This paper describes a study that sought to develop an instructional model using the most recent developments in technology as a means to enhance student learning.

In line with these ideas, The Dynamic Interactive Visualisation Tool in teaching C (DIVTIC) was designed as an interactive learning tool for introductory programming to help students to learn programming through the provision of a raft of useful resources supported by a tool to visualize and conceptualize programming constructs (Chansilp and Oliver, 2002; Chansilp and Oliver, 2004). A study of DIVTIC was implemented with learners and data gathered to explore their levels of use of the various tools and their degree of acceptance of the tool as a support for learning. The results from the study demonstrated that students made significant use of the tool and its various elements as a support for their learning. The results revealed the successful use of the animation tool as a strong online support tool for the teaching of this subject and as an appropriate learning support tool choice.

After the promising results of DIVTIC, the conceptual framework of the Dynamic Interactive Visualisation Tool in teaching Data Structure (DIVTIDS) was designed (Chansilp and Mukviboonchai, 2004). The foundation of DIVTIDS was designed around DIVTIC. However, some functionality had been adapted to be appropriate to the nature of the data structure course. Following the designed conceptual framework of DIVTIDS, the design and development of Dynamic Interactive Visualisation Tool in teaching Data Structure (DIVTIDS) had been conducted and presented (Chansilp and Mukviboonchai, 2005). The development of DIVTIDS was continued and supported by the 2006 Endeavour Australia Cheung Kong Awards and Department of Education, Science and Teaching, Australian government and given a more appropriate name, VIDSAA, Visualisation In Data Structure And Algorithms.

LEARNING OBJECTS

The development of computer-based resources for learning is very much guided these days by issues of sustainability and reuse. Whereas previously resources tended to be developed for use in particular settings, there are now strategies that can be adopted that provide increased opportunities for resources to be useful to users outside institutional contexts. The design of the learning resources within the VIDSAA environment sought to gain the opportunities from what has been discovered about the design and use of learning objects (Downes, 2000).

Current work with learning objects is seeking to explore and provide enabling systems and processes to enable teachers creating learning environments to be able to discover and locate online resources that can be seamlessly incorporated into learning environments. When one

examines the nature of e-learning and its use in educational settings, there are many factors that potentially limit a number of the goals and aims of the learning object movement (Polsani, 2003). For example:

- Learning resources come in a huge variety of forms and sizes;
- Most e-learning resources are developed and built for personal and local use without regard for reuse beyond the immediate context;
- They are built from a variety of technologies and in a variety of architectures which tend to tie them to particular platforms and operating systems;
- The resources have often been designed for use in a single setting, with hard links and connections that cannot be easily disconnected if the materials are to be used elsewhere;
- The resources contain references and descriptions from the local setting which could be out of place if the materials were reused.

VIDSAA

The development of the interactive resources in VIDSAA as learning objects to support student learning appeared as a useful design strategy that could heighten their usefulness and capacity to impact. Their development as stand-alone resources with a capacity for interoperability within existing delivery systems was seen as an important outcome. At the same time, it was evident that by considering in their design and development such aspects as their accessibility and interoperability would create flexibility in their capacity for reuse.

In planning the form of the various resources, a number of the issues associated with the design and development of learning objects guided many of the decisions taken in relation to the form and structure that was used. For example: the granularity (Duncan, 2003) was influenced by the opportunity to provide discrete resources for the various data structures and their algorithmic process. A consistent grain size was planned to facilitate the development of resources across the broad scope of data structures; the instructional form of each resource was planned to ensure there was a degree of neutrality within the instructional support and context to create flexibility in how and by whom the resources could be used (Friesen, 2004); the resources were planned as stand-alone entities each of which could be used independently of others and any overarching structure to facilitate reuse (Koper, 2003); the resources were to be designed as Flash animations as a means of ensuring their interoperability and capability for implementation within a broad range of courseware management systems (Koppi, 2005); and the use of the Flash format provided the opportunity to develop appropriate metadata descriptors which could be attached to each resource to aid discovery and reuse (Friesen, 2002) and provided opportunities for subsequent customisation and editing as part of the reuse process (Friesen, 2004).

THE DESIGN AND DEVELOPMENT OF VIDSAA

The learning objects in VIDSAA system comprised the major data structure forms including Recursion, Lists, Stacks, Queues, Trees, Sorting & Searching and Graph that are usually covered in an introduction course in Data Structures and Algorithms. There are 55 animations in total. The objects were designed as interactive animations each with three discrete components:

Overview: used to present information describing its topic (Figure 1), Algorithm and Source Code: used to present the algorithm and source code used to solve that problem (Figure 2), and Graphical Representation: used to show graphically on how the algorithm or source code is used (Figure 3).

The overview within each animation provides an interactive visual representation of the algorithm and/or data structure which the student can control. The use of play, stop, forward and backward buttons enables the student to step through the processing and to observe the changes to the variables and the relevant data. This animation provides the student with an overview of the process involved and is intended to help learners to conceptualise how the algorithm/data structure

functions. As the student plays the animation, the variables and data change enabling the student to observe what is typically concealed and hidden from view (Figure 1).

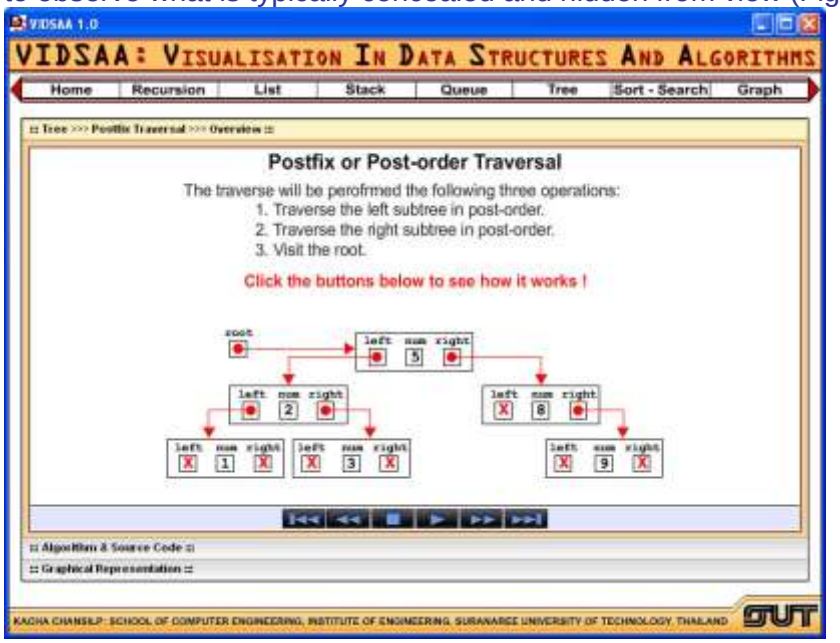

Figure 1: Data Structure Overview

The second element of VIDSAA displays the actual algorithm and source code. The algorithm is presented in a pseudocode form that highlights the variables and control structures. The logic that controls the algorithm is displayed and linked to the relevant lines of the source code. This element enables the students to examine the algorithmic processes and to see how these are translated into the actual programming code (Figure 2).

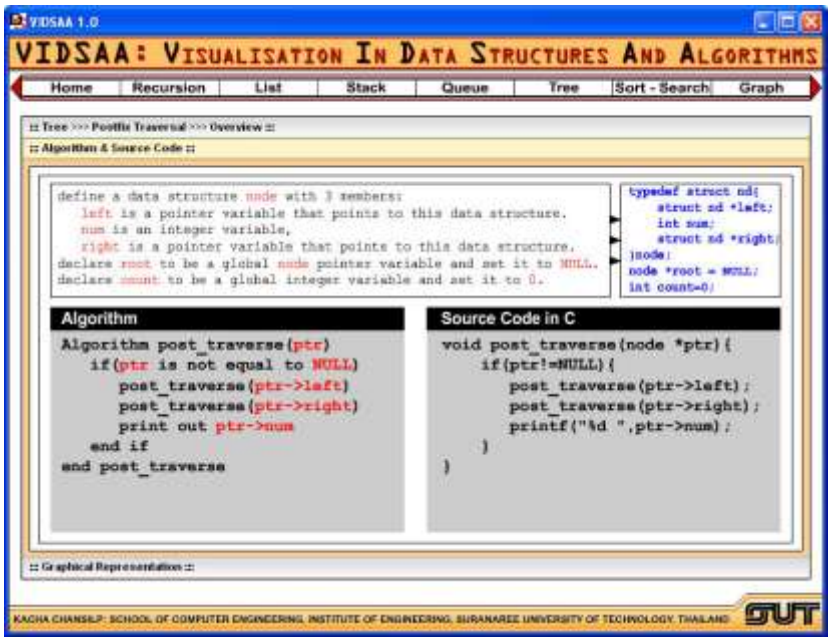

Figure 2: Data Structure Showing Algorithm and Source Code

The final element in VIDSAA is a representation which links the source code to a visual representation of the processing. The linking is provided in the form of a communication which enables the student to walkthrough the code from the commencement to the completion and to observe how the algorithm carries out the intended function (Figure 3).

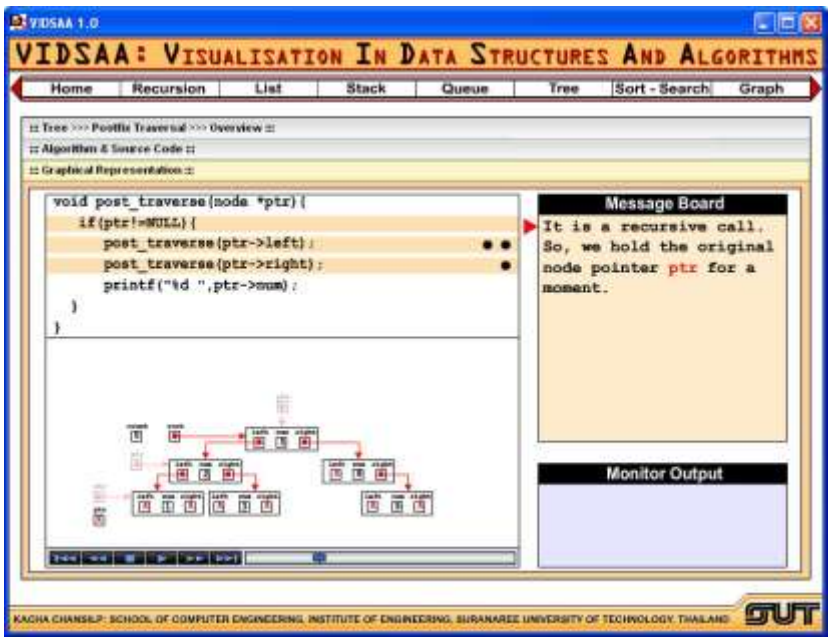

Figure 3: Data Structure Graphical Representation

To control each animation, learners use six static buttons, Play, Step-Backward, Step-Forward, Stop/Pause, Go to the End, and Go to the Beginning buttons, one slider bar and one dynamic slider button (Figure 4) that can be dragged into any position on the slider bar to see any particular spot of the animation. The slider allows students to control the animation process. It works the same way as the video controller buttons. This feature was intended to enable students to pause and think before watching a further step of the animation and this was intended to provide an opportunity for students to become active learners.

VIDSAA was planned to contain with all necessary elements needed to support normal classroom activity, supporting both teacher and student use. The animations can be used by the teacher to demonstrate and explain algorithms and data structures as part of an instructional presentation. As well as this, VIDSAA was designed with a capacity for installation onto a student's computer for use outside classroom activities as part of their independent learning..

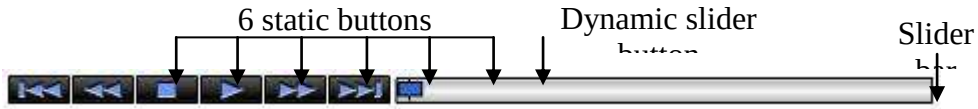

Figure 4: The control buttons

A SAMPLE INTERACTION

The following example demonstrates how students are able to use VIDSAA to develop their understanding of algorithms and data structures. Consider recursion, a data structure which many students find difficult to understand and apply. Figure 5 shows the flowchart from the learning object designed to demonstrate recursion. It provides a walkthrough on how to find the factorial value of a given number, 5, by using a function called recursive_factorial() and sending 5 as an argument.

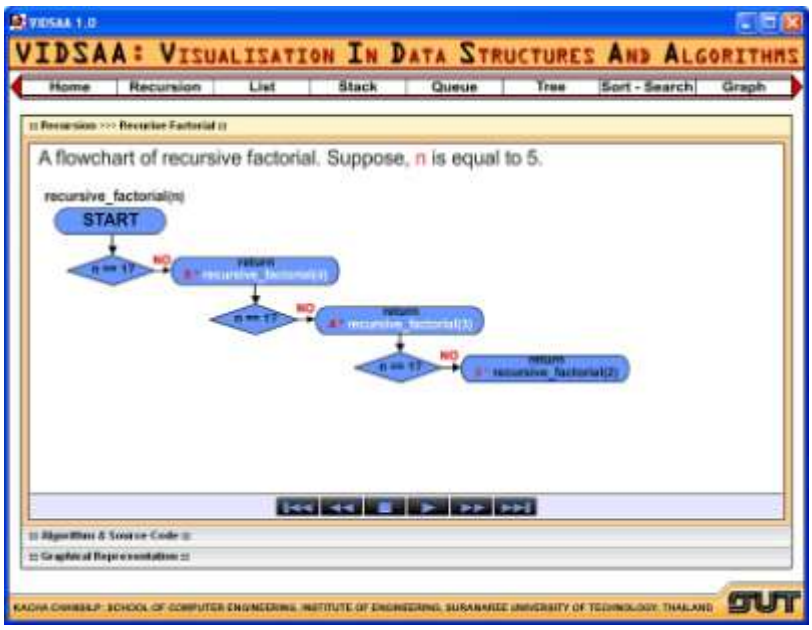

Figure 5: A flowchart of recursive factorial

Students can visualise and control the animation step-by-step on how the problem would be solved. The first step is to compare that given number, 5, to 1. If it is equal to 1, then return 1 to the caller, otherwise return that number multiply by function recursive_factorial (given number – 1), that is 5*factorial(4). The flowchart builds as the algorithm proceeds and the various comparisons, decisions and steps are followed. The repetition demonstrates how recursion enables calculations to be performed until a certain condition is met, at which stage the processing. The step is repeated until the argument is equal to 1 (Figure 6). Then, the value would be returned to the caller recursively.

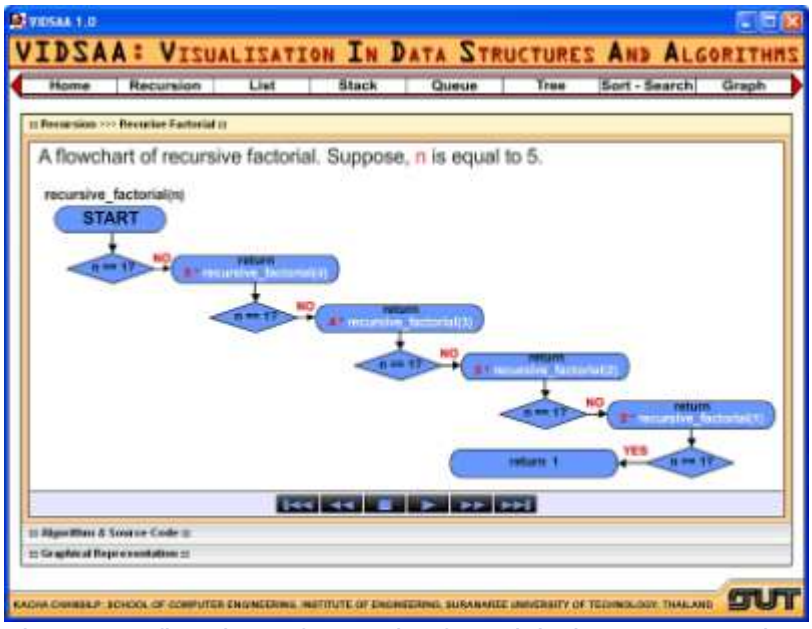

Figure 6: A flowchart of recursive factorial when argument is equal to 1

The interactions have been designed to give students full control over the animation. If at any stage the student wishes to stop the algorithm and to retrace the steps, the various controls facilitate this. The animation stops when the processing is complete and the final values are returned (Figure 7).

Figure 7 shows the returned values to the caller recursively back to the beginning of the function. The returned value would be computed and returned as the result to the caller. The various elements in VIDSAA operate in a consistent fashion across the broad range of data structures and algorithms usually contained in an introductory programming course.

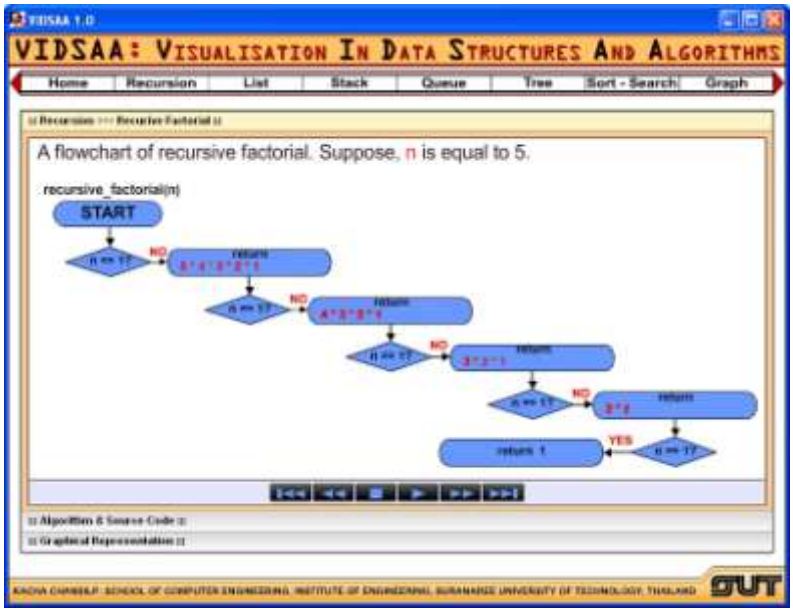

Figure 7: A flowchart of recursive factorial demonstrates the returned values

RESEARCH PLANS

There are many factors which can influence the success of learning objects as supports for student learning in university settings. As independent learning resources, students will often need to be motivated and encouraged to use them. Also, it is unclear as to what forms of learning support best facilitate their use. For example, is it best for teachers to set independent tasks for the students to complete or to simply encourage student exploration and inquiry? It will be important in the process of implementing these learning objects to explore what are the optimal strategies for encouraging and facilitating student use of them.

The question of student access is also an issue which can influence student use. If the learning objects are stored on servers, students will need to be reminded as to where they are located and what resources are available. On the other hand, if the learning objects are provided for students on their own computers, the increased accessibility might encourage their use. Understanding how best to provide access to resources will be an important issue to address.

A research project is now underway which is exploring implementation strategies and learning outcomes from the use of VISDAA in a large university in Thailand. The research involves the implementation of the learning objects as resources for students to access as part of their learning experience. The research is exploring such outcomes as: the scope and extent of student use of the resources; students' patterns of usage of the resources and the way in which the interactivity is used to support learning; factors influencing students' use of the resources; and teaching strategies that encourage and support students' independent use of the resources.

It is intended to use the findings from this research to develop guidelines to support the further development of learning objects of this form and to inform and guide teachers in optimal strategies for employing the learning objects in their teaching processes.

SUMMARY AND CONCLUSIONS

The paper has described the design and development of a set of learning objects, VIDSAA, Visualisation In Data Structure And Algorithms, intended to support the learning of data structures and their algorithms. The project recognized the difficulties faced by novice programmers and the opportunities provided by new learning technologies. Developed as learning objects, the resources have been designed to support student-centred learning across a variety of settings and learning platforms. The innovative elements of the project include the various forms of interactivity and engagement provided to support student learning derived from previous research of the authors.

The project is now exploring issues associated with the implementation of the resources to discover strategies that teachers can use to maximize their learning potential. Areas being explored include strategies to encourage student-centred activities and interaction patterns that maximize the learning potential. The outcomes from the study will provide information that can guide not only the implementation of these resources but also the design, development and application of resources as learning objects supporting learning in other conceptually demanding subjects and disciplines.

REFERENCES

Chandler, P. and Sweller, J. (1991). Cognitive load theory and the format of instruction. Cognition and Instruction, 8(4), 293-332.

Chandler, P. and Sweller, J. (1992). The split-attention effect as a factor in the design of instruction. British Journal of Educational Psychology, 62(2), 233-246.

Chandler, P. and Sweller, J. (1996). Cognitive load while learning to use a computer program. Applied Cognitive Psychology, 10(2), 151-170.

Chansilp, K. and Mukviboonchai, S. (2004). The conceptual framework of dynamic interactive visualisation tool in teaching data structure (DIVTIDS). EDU-COM 2004: New Challenges for Sustainability and Growth in Higher Education, Khon Kaen, Thailand.

Chansilp, K. and Mukviboonchai, S. (2005). The design and development of dynamic interactive visualisation tool in teaching data structure (DIVTIDS). The Seventh International Conferrence on Information Integration and Web-based Application & Services (iiWAS2005), Kuala Lumpur, Malaysia.

Chansilp, K. and Oliver, R. (2002). Using multimedia to develop students' programming concepts. EDU-Com 2002: Higher Education without Borders Sustainable Development in Higher Education, Khon Kaen, Thailand.

Chansilp, K. and Oliver, R. (2004). Students' responses to the use of a multimedia tool for learning computer programming. Ed-Media 2004: World Conference on Educational Multimedia, Hypermedia & Telecommunications, Lugano, Switzerland.

Downes, S. (2000). Learning Objects. Retrieved June 2001, from http://www.atl.ualberta.ca/downes/namwb/column000523_1.htm

du Boulay, B. (1986). Some difficulties of learning to program. Journal of Educational Computing Research, 2(1), 57-73.

Duncan, C. (2003). Granularization. In A. Littlejohn (Ed.), Reusing Online Resources: A Sustainable Approach To E-Learning (pp. 12-19). London: Kogan Page.

Friesen, N. (2004). Learning objects and standards: Pedagogical neutrality and engagement. Retrieved 15 November 2004, from www.learningspaces.org/ n/papers/pedagogical_neutrality.pdf

Kann, C., Lindeman, R. W., and Heller, R. (1997). Integrating algorithm animation into a learning environment. Computers and Education, 28(4), 223-228.

Karsten, R., and Kaparthi, S. (1998). Using dynamic explanation to enhance novice programming instruction via the World Wide Web. Computers and Education, 30(3/4), 195-201.

Koper, R. (2003). Combining reusable learning resources and services with pedagogical purposeful units of learning. In A. Littlejohn (Ed.), Reusing Online Resources: A Sustainable Approach to e-learning (pp. 46-59). London: Kogan Page.

Koppi, T., Bogle, L., and Bogle, M. (2005). Learning objects, repositories, sharing and reusability. Open Learning, 20(1), 83-91.

Lischner, R. (2000). Programming Language And Tools For Deep Learning. Retrieved August 22, 2000, from http://www.cs.utexas.edu/users/csed/doc_consortium/DC99/lischner-abstract.html

Mulholland, P. and Eisenstadt, M. (Eds.) (1998) Using Software To Teach Computer Programming: Past, Present And Future. MIT Press, Cambridge, Massachusetts.

Polsani, P. (2003). Use and abuse of reusable learning objects. Journal of Digital Information, 3(4). Retrieved 11 September 2006, from http://jodi.tamu.edu/Articles/v03/i04/Polsani/

Rowe, G., and Thorburn, G. (1999, August). Evaluate of VINCE - a tool for teaching introductory programming. Paper presented at the 7th Annual Conference on the Teaching of Computing, University of Ulster, Northern Ireland.

Sweller, J. (1998). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12, 257-285.