2003

Using multimedia to develop students' programming concepts

Kacha Chansilp
Edith Cowan University

Ron Oliver
Edith Cowan University

Chansilp, Kacha. Edith Cowan University, Australia. *Using Multimedia to Develop Students' Programming Concepts.*

K Chansilp[1] and R Oliver[2]

[1]School of Communications and Multimedia
Edith Cowan University, Australia,
E-mail: k.chansilp@ecu.edu.au

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

ABSTRACT

The growing use of computers and computer technology is creating a worldwide need for more programmers and computer professionals. The increasing complexity of programming languages and applications is demanding higher skills sets from the programmers who are being trained. Kann et al. (1997) suggest that the graphic representation of algorithms used in most textbooks are too abstract and insufficient for learners to develop the logical thinking required in computer science courses. Many programming students have problems due to a lack of understanding of conceptual and mental models (Soloway et al., 1981).

This paper describes the development of an alternative teaching approach, based on constructivist learning principles, and multimedia technologies. The Dynamic Interactive Visualisation Tool in teaching C (DIVTIC), uses multimedia and visual imagery to provide learners with a step-by-step representation of program executions in the C language as a means to enhance their understanding. DIVTIC was designed around constructivist principles, and combines collaborative and visualisation learning strategies with use of the Internet and the World Wide Web to support the learning of programming.

This paper will describe the conceptual framework supporting the design of DIVTIC and will report on a study which sought to explore the effectiveness of its use among a cohort of students studying introductory programming at Suranaree University of Technology (SUT) in Thailand. The paper will describe how the students used DIVTIC and will discuss how this use was able to support and encourage their learning.

INTRODUCTION

As technology is growing rapidly, interactive multimedia technology is being used more and more in the educational sector. To keep pace with this rapid change, researchers need to optimise the use of new technologies in teaching and learning environments. Thus, instructional materials and tools are being produced by incorporating updated software with such advanced delivery technologies as the Internet (e.g., Oliver et al., 1996; Rowe and Thorburn, 1999; Warendorf, 1997; Yoo, 1998).

The teaching and learning of any programming language is a difficult activity. Conventional instruction may not always be appropriate to help students in developing semantic knowledge (Oliver and Malone, 1993). There have been many efforts to develop tools for the teaching and learning of programming languages (e.g., Daly, 1999; Jehng and Chan, 1998; Rowe and Thorburn, 1999; Smith and Webb, 1998). However, this still remains a problem in many institutions (Carter and Jenkins, 1999). The reasons for this are that the development of new languages make current resources outdated, the focus of the learning is on syntax rather than on semantics and concepts, and the difficulty in visualising of program executions.

New programming languages are more complex because they have more advanced features. For example, for many years, Pascal was the major introductory computer programming language and was the most popular first language for teaching students in the nineties (Brilliant and Wiseman, 1996).

However, Pascal is no longer used in industry (Hubbard, 1996) and many institutions have now switched from Pascal to C. The problems experienced by novice programmers learning a procedural language such as Pascal as their first programming language are increased when learning C (Hubbard, 1996; Smith and Webb; 1998). The C language is too difficult for many novices to acquire on their own (Johnson, 1995).

BACKGROUND

Computer programming courses are more difficult and time consuming than other courses for the majority of students (Hagan and Lowder, 1996). Usually, novices have different background knowledge before entering an introductory computer programming course. Consequently, a class may contain a range of proficiency levels. Wilcocks and Sanders (1994) suggest the use of a computer-aided dynamic program animator to support the instructors of weaker student programmers on those who may be too shy to ask questions and on those who are difficult to assist when classes are large.

Generally, students differ in their ability to understand material that is very abstract and difficult to visualise. Previous research has proposed ways to improve instructional materials and therefore student outcomes. For example, instruction can incorporate a dynamic explanation tool to help students visualise each step in program execution (Karsten and Kaparthi, 1998; Rowe and Thorburn, 1999; Lischner, 2000). 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.

OPPORTUNITIES OF MULTIMEDIA

Students in science majors, especially in the computing field, need to learn valid basic concepts during their introductory computing courses as these form a strong background for more advanced programming courses in their college curriculum (Herrmann and Popyack, 1994). At the moment, current teaching is based on textbooks and this does not always work well. Many students who finish introductory classes, do not know what they are doing. Kann et al. (1997) suggest that the graphic representation of algorithms used in most textbooks are abstract visualisations and are not sufficient for learners to develop the logical thinking required in computer science courses. Students' problems are mostly based on a lack of understanding of conceptual and mental models (Soloway et al., 1981).

This study explored the opportunity for creating a technology based learning model that used visualisation to provide student activities for building their understanding of the programming process. The model was based on the learning principles necessary for learning successful programming; it was derived from the literature that can be used to inform and guide the design of the learning environments. Methods of visualisation, collaboration, constructivism, and student-centred learning were used to explore how the design might be used to best benefit teaching and learning in introductory computer programming. This model also provides a benefit for instructors who may want to move from traditional instruction methods to using technology as a teaching medium. The study sought to explore strategies to use technology, not only to help students to learn effectively, but also to:

- prepare students with a strong background in computer programming,
- reduce learning time,
- save teacher consultation time, and
- engage students in student-centred activities.

THE DESIGN OF AN INTERACTIVE LEARNING SETTING

The study involved a design of an interactive learning model, Dynamic Interactive Visualisation Tool in Teaching C (DIVTIC). The design of DIVTIC was based on the learning principles necessary for successful programming learning; it was derived from the literature that has been used to inform and guide the design of the learning environments. DIVTIC is composed of 8 sections and its characteristic features, along with the learning principle, are shown in Table 1. DIVTIC was designed

to be more visually explicit than the existing systems (e.g., Rowe and Thorburn, 1999). DIVTIC shows phrases in each line and employs a combination of complementary tools that encourage students to be active learners.

Table 1: The Relationship between DIVTIC Characteristic and Learning Principle

DIVTIC Characteristic	Learning Principle
Syllabus/Lecture Notes	Student-centred resource
Computer Structure	Student-centred resource
Animated Examples	Visualisation and Constructivism
'C' Compiler	Student-centred resource
'C' WebBoard	Collaboration
Self-Evaluation	Constructivism
FAQ Pool	Constructivism
'C' References & Links	Constructivism

The elements of DIVTIC are described as follows:

- **Syllabus/Lecture Notes:** There are six sections including syllabus, textbook, lecture notes, laboratory, weekly task, and sample sections. The syllabus section includes the information on topics taught during the semester, the laboratory test date, midterm and final test dates, and relevant course textbooks. The textbook section includes a revised version of the previous lecture notes. The lecture notes section contains power point slides used by the instructor in the lectures. The laboratory section is where all students who are registered in the course can download the weekly laboratory problem ahead of time. The weekly task section is designed to contain weekly activities that the students can be required to do at the beginning of each laboratory session. The sample section is designed to contain a sample of previous midterm and final examinations.
- **Computer Structure:** This section was designed to offer an overview of the basic structure of a computer and contains 12 different parts of hardware including a monitor, central processing unit (CPU), CD-ROM, hard disk, floppy disk, mouse, computer system, scanner, printer, modem, digital camera and zip drive. Each part is given a definition and description.
- **Animated Examples:** This section contains 10 chapters. Each chapter is comprised of three to six animated examples. The students move the mouse over each chapter to reveal a submenu containing some animated examples.
- **'C' compiler:** This section was designed to teach the users how to use C compiler step-by-step.
- **'C' WebBoard:** This section was designed to permit students to have asynchronous discussion so that they would share ideas, information, questions and suggestions by reading and posting the messages at whatever hour of the day they are most productive. The instructor can also visit the WebBoard page daily to reply to those messages. This ensures that all messages have been answered by either a peer or the instructor himself.
- **Self-Evaluation:** In this section, students test their understanding by clicking on each topic. The students can then do the self-assessment test by selecting an answer. JavaScript was used to create an immediate feedback feature that pops-up right after an answer has been selected.
- **FAQ Pool:** This section groups frequently asked questions into related chapters.
- **'C' References & Links:** This section is a collection of the URLs containing some useful information about C language.

Access to the various features is gained through an opening menu screen (Figure 1).

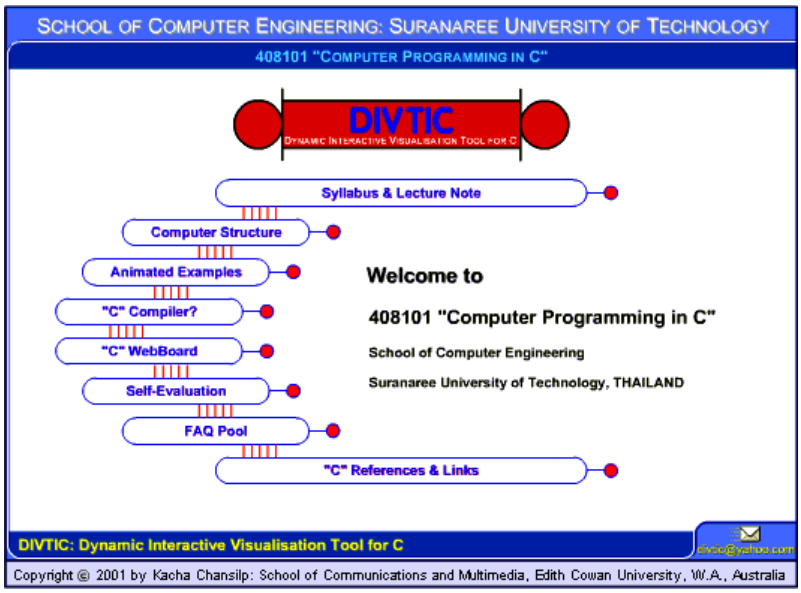

Figure 1: DIVTIC's homepage

As the Animated Examples section was the most important section, there is a need to describe its features in more detail. There are four panels in each animated example including C Source Code, Message Board, Monitor Output, and Memory Map panels (see Figure 2).

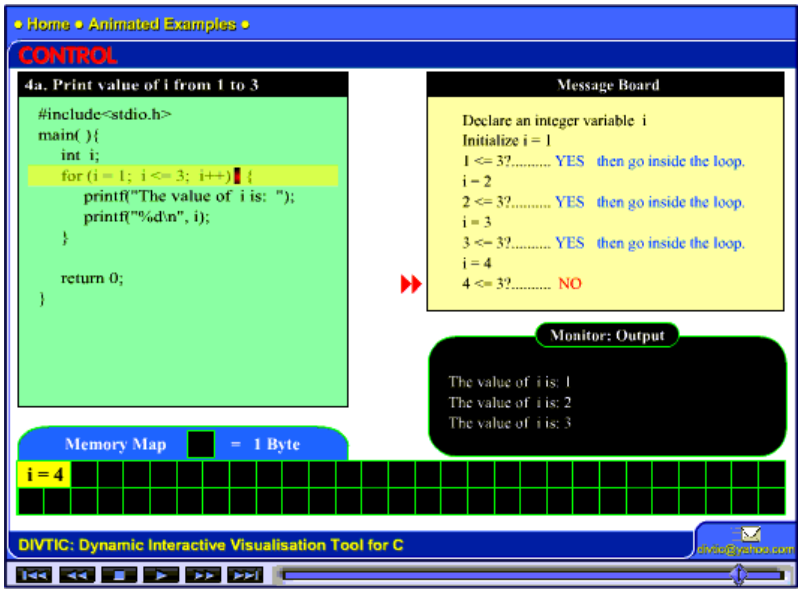

Figure 2: A sample page of an animated example

The top left-hand side is a panel for a given problem to be displayed with an icon at the bottom called 'C Source Code'. When students click on this icon, the given problem then changes to the C source code as shown in Figure 2. This feature was incorporated because it would allow the user to pause and think about the given problem and to try to write the code before clicking on the icon to see the result. This would encourage the students to be active learners. The bottom of the C Source Code panel was designed to include Control Menu buttons which are used to control the animation. These work the same way as the video controller buttons. Six static buttons, one slider bar and one dynamic slider button can be dragged into any position on the slider bar to see any particular spot of the animation. The top right-hand side has another panel called 'Message Board'. Its purpose is to display dynamic messages at any specific time when the marker in the Source Code panel runs pass any particular commands. The panel below the C Source Code and Message Board, called the 'Monitor Output',

synchronously demonstrates the real-time output. Another panel located under the Monitor Output panel is called the 'Memory Map'. This Memory Map panel helps the users develop mental models of how a computer stores data and its values. When the marker runs through the declaration section, the equivalent section in the memory map is highlighted. The number of boxes is dependent on the type of that particular variable and is individually assigned to that variable.

Other sections of the environment include the Syllabus/Lecture Notes, Computer Structure, 'C' Compiler, 'C' WebBoard, Self-Evaluation, FAQ Pool, and 'C' References & Links. Figure 3 shows a sample screen from the Self-Evaluation sections. The forms in the Self-Evaluation section use primarily multiple choice questions with immediate computer-generated feedback for learners.

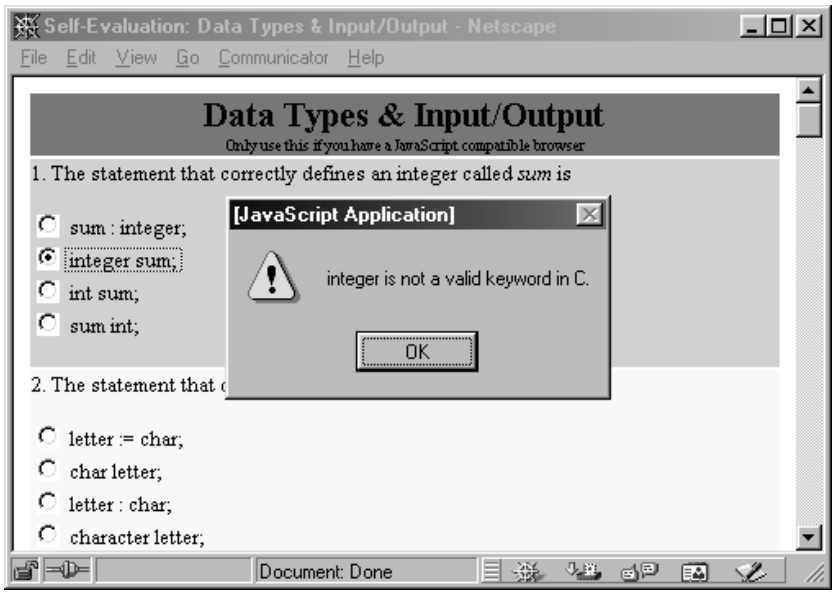

Figure 3: A sample screen from the Self-Evaluation section

EXPLORING HOW STUDENTS USED DIVTIC AND THE IMPACT ON LEARNING

Once DIVTIC was built and tested, a comprehensive study was planned to explore how students used DIVTIC and the extent an interactive visualisation process could influence learning outcomes. The study sought to explore such aspects as:

- Which parts of DIVTIC would students use and for how long?
- What strategies would students use with DIVTIC?
- What factors would influence students' use of DIVTIC?
- What attitudes would students have towards DIVTIC e.g., user-friendly, enjoyable, valuable, boring, useability, and clarity?
- How would this process influence students' performance in programming?
- To what extent would this way of learning vary between students?
- To what extent would this process influence students' high-order thinking, confidence and motivation in learning introductory programming? and
- What factors would influence students' achievement with this process e.g., time, attitude, collaboration, experience?

This study was conducted in Thailand at Suranaree University of Technology (SUT), which has three trimesters in one year, during the second trimester of 2001 from September to December for the total of 13 weeks. Approximately 500 undergraduate engineering students enrolled in Computer Programming 408101, 100 took part in the study. The conventional teaching practice of 'Computer Programming 408101' included a two-hour lecture in a lecture theatre hall and a two-hour laboratory session which was scheduled after the lecture. The laboratory was divided into 10 sessions. The students registered into one of these 10 sessions on a first come-first serve basis. There were 50

students and two tutors in each section. There were four computer laboratories, each with 60 networked personal computer systems.

The full comprehensive study used both quantitative and qualitative methods throughout the whole 13 weeks. Table 2 shows the type of instruments used and the data collected during the entire study.

Table 2: Types of Instruments used and Time to be collected

Instrument	Group		Week												
	C	E	1	2	3	4	5	6	7	8	9	10	11	12	13
Weekly Task		x		x	x	x	x	x		x	x	x		x	
Researcher's Observation		x		x	x	x	x	x		x	x	x		x	
Tutors' Observation		x		x	x	x	x	x		x	x	x		x	
Self-administered Questionnaire Form		x						x				x			
Subject Semi-structured Interview Form		x		x	x	x	x	x		x	x	x			
Tutors and Instructor Semi-structured Interview Form		x							x				x		
DIVTIC Log File		x		x	x	x	x	x	x	x	x	x	x	x	
Screen Video Capture Software		x			x	x	x	x		x	x	x		x	
WebBoard		x		x	x	x	x	x	x	x	x	x	x	x	
Email		x		x	x	x	x	x	x	x	x	x	x	x	
Laboratory Test 1	x	x							x						
Midterm Examination	x	x							x						
Laboratory Test 2	x	x											x		
Final Examination	x	x													x

ORGANISATION OF RESOURCES

The study was designed to use one group of 50 students as a control group (Group C) and another of 50 students as an experimental group (Group E) with the same tutors for both groups. Convenience sampling was used to select subjects by matching the GPA manually in choosing sample groups. Therefore, both groups were comprised of up to 50 students ranging from low to high GPA. Each group was divided into three different levels according to GPA: low—less than 1.78; average—1.78 to 2.22; and high—above 2.22. Group C, for example, included C1, C2, and C3 which referred to low, average, and high GPA respectively, and Group E was comprised of E1, E2, and E3 which also referred to low, average, and high GPA respectively. Dividing the GPA into three levels helped the study determine the different achievements of each level. Finally, both groups were perfectly matched one-by-one based on GPA as well as gender. Each group comprised approximately the same number of students based on gender as shown in the following table.

Table 3: Matching number of students in both groups based on gender

Gender	Low GPA (1.11 – 1.72)		Average GPA (1.78 – 2.22)		High GPA (2.25 – 3.36)	
	C1	*E1*	C2	E2	C3	E3
Male	8	8	9	8	8	9
Female	8	8	8	9	9	8

A quasi-experimental design was used in this study since the subjects were not randomised. The experimental group used DIVTIC throughout the trimester while the control group did not. The following table shows that both control and experimental groups were treated in the same manner

except that the experimental group was provided with access in using the DIVTIC system as a supplemental tool in helping their learning.

Table 4: Quasi-experimental design without pretest in both groups

Group	DIVTIC System	Lab Test 1 (10%)	Midterm (30%)	Lab Test 2 (10%)	Final (50%)
Experimental (Group E)	x	x	x	x	x
Control (Group C)	o	x	x	x	x

The DIVTIC system was uploaded onto a Linux server located in the instructor's office. Each subject in Group E was assigned a personal password to use with his/her identification to log into the DIVTIC system. The log in process was demonstrated by the researcher, who also explained how to navigate and highlighted the various features available in the DIVTIC system in the first laboratory session. At the beginning of each laboratory session, starting from week 2 onward, the experimental group was given a weekly task relating to the topic taught in the lecture (see Figure 4). After the task was completed, the subjects logged into the DIVTIC system and ran a specific animated example to check their answers within 30 to 45 minutes. They used the remaining time to complete the instructor's weekly problem. This was designed to engage all subjects in completing the weekly task before logging into the DIVTIC system. On the other hand, the control group used the whole two hours to complete the instructor's weekly problem.

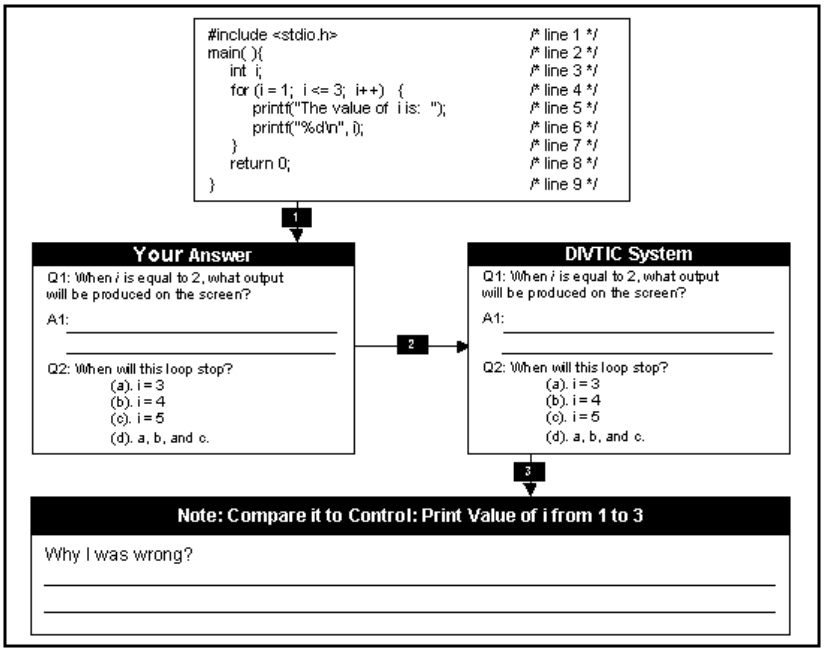

Figure 4: A sample of the weekly task

FINDINGS

This paper reports on the findings from the study in relation to the question of whether use of DIVTIC could be seen to enhance students' programming performance. In particular, it explores evidence of how the use of DIVTIC was seen to enhance student learning by comparing the results of the control and experimental groups in laboratory test 1, midterm examination, laboratory test 2, and the final examination.

There were 17 subjects in Group C and 11 subjects in Group E who withdrew from the course. Thus, there were only 72 subjects (Group C = 33; Group E = 39) participated in this study. Table 5 shows the number of the participated subjects in each group and level. It reveals that the majority of the

subjects who withdrew from the course were those with a low GPA in both groups (C1 = 56.25%, E1 = 56.25%). However, there were no subjects with a high GPA in the experimental group who withdrew from the course. This suggests that students with a low GPA had a more difficult time in learning to perceive and construct their knowledge in learning introductory computer programming than students with an average or high GPA.

Table 5: The number of participated subjects and percentage of withdrawal in each group and level

Gender	GPA 1.11 – 1.72		GPA 1.78 – 2.22		GPA 2.25 – 3.36	
	C1	E1	C2	E2	C3	E3
Participated Subjects	7	7	14	15	12	17
Withdrawal Percentage	56.25%	56.25%	17.65%	11.76%	29.41%	0%

All the scores for each student in both groups including the laboratory test 1(10%), midterm examination (30%), laboratory test 2 (10%), and final examination (50%) were entered into an excel spreadsheet for analysis with SPSS.

Students with Low GPA

Table 6 shows the mean and standard deviation of each test of the two groups in level 1 (Low GPA 1.11 – 1.72). It reveals that students in Group E1, who used DIVTIC, had a higher mean in each test than those in Group C1. The t-test showed that the difference between the means of the two groups in laboratory test 1(2.28) and laboratory test 2 (1.64) was not large enough to reach statistical significance. However, there were obviously relatively large differences between means in the midterm examination, (3.14) with $t(8.48) = 2.20$, $p < 0.05$, and the final examination (6.07), and there was a significant difference between the means with $t(12) = 3.46$, $p < 0.05$. It appeared that the students had not been using DIVTIC long enough by laboratory test 1 for the tool to make a difference. However, continued use throughout the trimester appeared to contribute to students' programming performance.

Table 6: The mean and standard deviation of each test between the experimental and control groups in level 1

Group	No.	Lab Test 1		Midterm		Lab Test 2		Final	
		M	SD	M	SD	M	SD	M	SD
Experimental 1 (E1)	7	7.3571	2.60951	16.7143*	3.42609	6.3571	2.52841	32.0257*	3.69253
Control 1 (C1)	7	5.0714	2.11007	13.5714*	1.59239	4.7143	1.86764	25.9529*	2.82138

* $p < 0.05$

Students with Average GPA

Table 7 shows the mean and standard deviation of each test of the two groups in level 2 (Average GPA 1.78 – 2.22). Students in Group C2 had a slightly higher mean than Group E2 in laboratory test 1 (0.04), in the midterm examination (1.96), and in the laboratory test 2 (0.35). However, students in Group C2 had a significantly higher mean in the final examination (4.31). The use of DIVTIC appeared to help the students in the early stage of the course, as evidenced by the laboratory test 1, but did not have the expected influence overall. In fact, other students who did not use DIVTIC scored significantly better. A possible explanation for this is that use of DIVTIC is time consuming and takes away from time spent in solving actual problems and tasks. This suggests that for the more able students, who may have already understood programming principles, use of DIVTIC may have actually impeded learning programming by providing less time for problem solving.

Table 7: The mean and standard deviation of each test between the experimental and control groups in level 2

Group	No.	Lab Test 1		Midterm		Lab Test 2		Final	
		M	SD	M	SD	M	SD	M	SD
Experimental 2 (E2)	15	5.4667	2.72860	15.4000	2.54390	5.4000	1.89171	28.6667*	5.94108
Control 2 (C2)	14	5.4286	2.53329	17.3571	3.04093	5.7500	2.59993	32.9757*	3.07900

* $p < 0.05$

Students with High GPA

Table 8 shows the mean and standard deviation of each test of the two groups in level 3 (High GPA 2.25 – 3.36). It reveals that students in Group E3 had less means than Group C3 in all tests. Students in Group C3 performed better in all the tests. These results were very similar to those achieved by students with an average GPA.

Table 8: The mean and standard deviation of each test between the experimental and control groups in level 3

Group	No.	Lab Test 1		Midterm		Lab Test 2		Final	
		M	SD	M	SD	M	SD	M	SD
Experimental 3 (E3)	17	5.5294	2.70110	16.0588*	2.89332	4.9118	3.33652	28.1382*	6.56616
Control 3 (C3)	12	5.8750	2.25756	18.6250*	3.56195	6.8333	2.30940	33.6817*	4.99322

* $p < 0.05$

The results suggest that DIVTIC appeared to help students with a low GPA to better understand a concept by providing visual imagery to help them, in the form of a step-by-step representation of program executions, assisting them to achieve a better outcome. On the other hand, the use of DIVTIC did not appear to help students with an average or high GPA. Possible reasons might be that DIVTIC did not provide an adequate challenge for them or that they did not spend as much time with DIVTIC as the others. These factors are planned explorations and will be announced through more detail in further papers to be written on DIVTIC.

SUMMARY AND CONCLUSIONS

This paper has described the design of DIVTIC, a tool incorporating constructivist principles, collaborative, and visualisation learning strategies with use of the Internet and the World Wide Web to support the learning of programming. The results from a quantitative study appear to support the notion that use of DIVTIC can assist novices in learning introductory computer programming. The results are interesting in that they clearly demonstrate the advantage of DIVTIC with students with a low GPA. The students from this level in the experimental group, significantly outscored their counterparts in the control group in the final test suggesting that DIVTIC was an important element in their learning outcomes. Further study is needed to establish precisely the design of elements of interactive multimedia which encourage and enable students with average or high GPA to achieve their ultimate outcomes. Furthermore, the two-hour laboratory session did not seem to be adequate. Novice students may need more time to finish both DIVTIC's weekly task and the instructor's weekly problem with tutor's assistance. In some cases, if the Internet connection possibly is a problem, then the DIVTIC system could be produced on a CD and given to all students. The study has suggested that DIVTIC is a valuable tool for some novice programmers and has encouraged further exploration and inquiry.

REFERENCES

Brilliant, S. S. and Wiseman, T. R. (1996). *The first programming paradigm and language dilemma.* Paper presented at SIGCSE'96. Philadelphia, USA, pp. 338-342.

Carter, J. and Jenkins, T. (1999). *Gender and programming: What's going on?* Paper presented at The 4th Annual SIGCSE/SIGCUE on Innovation and Technology in Computer Science Education. Krakow Poland, pp. 1-4.

Daly, C. (1999). *RoboProf and an introductory computer programming course.* Paper presented at The 4th Annual SIGCSE/SIGCUE on Innovation and Technology in Computer Science Education. Krakow Poland, pp. 155-158.

Hagan, D. and Lowder, J. (1996). *Use of the World Wide Web in introductory computer programming.* Paper presented at ASCILITE 1996. Adelaide, South Australia, pp. 247-257.

Herrmann, N. and Popyack, J. L. (1994). *An integrated, software-based approach to teaching introductory computer programming.* Paper presented at The Twenty-fifth Annual SIGCSE: Symposium on Computer Science Education. Phoenix, Arizona, USA, pp. 92-96.

Hubbard, D. (1996). *Education of computer programming,* [on-line]. Available: http://www.gl.umbc.edu/~schmitt/331S96/dhubba1/report.html [2000, 23 August].

Jehng, J. J. and Chan, T. (1998). Design computer support for collaborative visual learning in the domain of computer programming. *Computer in Human Behavior,* 14**,** pp. 429-448.

Johnson, L. F. (1995). Technical opinion: C in the first course considered harmful. *Communications of the ACM,* 38**,** pp. 99-101.

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

Karsten, R. and Kaparthi, S. (1998). Using dynamic explanation to enhance novice programming instruction via the WWW. *Computer and Education,* 30**,** pp. 195-201.

Lischner, R. (2000). *Programming language and tools for deep learning,* [on-line]. Available: http://www.cs.utexas.edu/users/csed/doc_consortium/DC99/lischner-abstract.html [2000, 22 August].

Oliver, R., Herrington, J. and Omari, A. (1996). *Creating effective instructional materials for the World Wide Web.* Paper presented at AusWeb96: The Second Australian WorldWideWeb Conference. Conrad Jupiters Hotel, Gold Coast, Queensland, Australia, pp. 485-492.

Oliver, R. and Malone, J. (1993). The influence of instruction and activity on the development of semantic programming knowledge. *Journal of Research on Computing in Education,* 25**,** pp. 521-533.

Rowe, G. and Thorburn, G. (1999). *Evaluate of VINCE - a tool for teaching introductory programming.* Paper presented at 7th Annual Conference on the Teaching of Computing. University of Ulster, Northern Ireland, pp. 99-102.

Smith, P. A. and Webb, G. I. (1998). *Overview of a low-level programming visualisation tool for novice C programmers.* Paper presented at ICCE'98: Global Education on the Net. Beijing, China, pp. 213-216.

Soloway, E., Ehrlich, K., Bonar, J. and Greenspan, J. (Eds.) (1981). *What do novices know about programming?* ABLEX Publishing Corporation, New Jersey.

Warendorf, K. (1997). *ADIS - an animated data structure intelligent tutoring system on the WWW*. Paper presented at International Conference on Information, Communications and Signal Processing ICICS'97. Singapore, pp. 944-947.

Wilcocks, D. and Sanders, I. (1994). Animating recursion as an aid to instruction. *Computer and Education,* 23**,** pp. 221-226.

Yoo, Y. D. (1998). *An interactive hypermedia learning system: LMM.* Paper presented at ICCE'98: Global Education on the Net. Beijing, China, pp. 504-510.