Creating Visualizations from Multimedia Building Blocks: A Simple Approach to Teaching Programming Concepts

Tanya Linden
tanya.linden@vu.edu.au
School of Management and Information Systems
Victoria University
Melbourne, Victoria, Australia

Reeva Lederman
reeva.lederman@unimelb.edu.au
Department of Information Systems
The University of Melbourne
Melbourne, Victoria, Australia

Abstract

Academics teaching programming are faced with the challenge of teaching dynamic concepts using static media. Despite multiple tools developed in the past to support learning of programming concepts, access to these tools is very limited and many educators have to create their own support materials and tools which is often time-consuming, complex and expensive. In this paper we share our experience of producing simple animations using widely and/or freely available software packages to develop small building blocks that are customizable and reusable and can be mixed and matched to meet the learning requirements of individual students. Positive student feedback from using these tools has encouraged the continued exploration of options for developing animations illustrating programming concepts.

Keywords: programming concepts, program visualization, multimedia repository

1. BACKGROUND

The majority of publications on teaching introductory programming concepts suggest that students experience difficulties learning the concepts and academics find teaching these programming concepts to novices challenging (Bennedsen, E.Caspersen, & Kölling, 2008; Jenkins, 2001; Matthews, Hin, & Choo, 2009; Prasad & Fielden, 2003; Rudder, Bernard, & Mohammed, 2007; Stone & Clark, 2011). Over years there has been a change in the programming languages being taught (e.g. from Pascal and C to Visual Basic and Java), more creative teaching approaches have evolved and technological developments have been applied, however academics are still experiencing the problems associated with teaching programming to novices (Hadjerrouit, 2008; Spronken-Smith & Harland, 2009). Researchers have investigated various reasons for the difficulties of learning programming concepts (Gomes & Mendes, 2007; Jenkins, 2002).

One of the challenges is teaching dynamic concepts using static materials (Gomes & Mendes,

2007). Teaching staff use whiteboards, static presentations and printed materials, such as notes and textbooks, in order to explain concepts such as program behavior, dynamic memory allocation or change of variables' values during program execution. Students learning these concepts often find it hard to imagine how each source code instruction is processed by the computer and what is happening inside RAM as program statements are executed. However, this understanding is important if students are to learn how to write programs.

Unfortunately, despite gaining sophistication and improved user friendliness over the years, modern programming environments are still not very friendly when it comes to use by novice programmers. From a teaching perspective one of the weaknesses of professional environments lies in the fact that they do not offer a beginner-friendly way of tracing program execution and seeing what is happening in RAM. So those academics who resolved to use visualization, i.e. application of visual technology to graphically illustrate abstract concepts (Naps, et al., 2003), are confronted with the problem: how do we provide a "window" into RAM? This paper proposes a solution to this question, a solution that can be implemented by any programming instructor without the difficulty of mastering a sophisticated development environment or a complex implementation curve.

The solution developed in this research is based on theories of active learning and adaptive learning. Active learning is defined as learning modes that "involve students in doing things and thinking about the things they are doing" (Bonwell & Eison, 1991, p.2). This theory suggests that students who play an active role in constructing knowledge will have both greater retention and greater learning enjoyment than students involved in passive pedagogical teaching approaches (Dufresne, Gerace, Leonard, Mestre, & Wenk, 1996). Approaches to adaptive learning recommend employing individualized learning programs adapted to student capabilities and preferred learning modes (Magoulas, Papanikolaou, & Grigoriadou, 2003).

Our solution applies these theories in creating student centered tools tailored to individual student abilities. The proposed tools development relies on the features supported by readily available software, such as easily modifiable PowerPoint animations, mp3 recording of explanations and screen capture. The software packages offering these capabilities are either available at most educational institutions or can be downloaded as freeware or evaluation copies from the Internet. Additionally, there is no difficult learning curve for developers as all the mentioned packages have a very friendly user interface.

2. VISUALIZATIONS IN TEACHING PROGRAMMING

Previous research has demonstrated that information visualization amplifies cognition (Card, Mackinlay, & Shneiderman, 1999). Since the 1980's there has been a growing interest in computer-supported visualization in artificial intelligence and computer graphics communities.

In programming two main approaches to visualization in support of teaching can be identified (Pears, et al., 2007; Smith & Webb, 2000):

- algorithm animation which aims at illustrating how an algorithm works and is largely programming language independent, and
- program visualization, which is programming language dependent and focuses on animating source code execution as well as showing changes in the variables states.

There is a strong belief among computer science educators that visualizations help in understanding programming concepts (Naps, et al., 2003), however there is little empirical evidence showing that beginner programmers benefit by using visualization tools. A thorough examination of the research literature reveals only one experimental study that compared understanding of programming concepts by students with access to a visualization environment and students without such access. This was conducted by Smith and Webb (2000). Their findings provide some preliminary support for what many educators already believe from their experience that visualization tools are of great help to learners.

Despite lack of significant empirical evidence on the advantages of using visualization tools in teaching introductory programming concepts (Hundhausen & Douglas, 2000), academics have developed a wide range of such tools that illustrate program execution line by line, for example Teaching Machine (Bruce-Lockhart & Norvell, 2000), AnimPascal (Satratzemi, Dagdilelis, & Evagelidis, 2001), VINCE (Rowe & Thorburn, 2000), OGRE (Milne & Rowe, 2004), a web-based tool (Donmez & Inceoglu, 2008). Bruce-Lockhart and Norvell (2000), as well as developing the software also produced video demonstra-

tions of program executions as supplementary materials to traditional lectures.

Despite this plethora of tools discussed in the literature, educators still don't have wide access to those tools. There are multiple reasons for that (Pears, et al., 2007). Most tools are developed as part of research projects. As a result development may stop as soon as either the working prototype meets the research objectives or research funding runs out. Developers move on to other projects while the tools still require fine tuning before they can be considered ready for distribution to other educators. Some tools are developed to meet the requirements of academics delivering specific courses. These tools are often not flexible enough to be adapted by other institutions or for other courses. Some academics experience "not invented here" syndrome. As a result despite multiple tools produced for similar purposes, every educator may need to 're-invent the wheel'. Even enthusiastic educators of programming admit that embedding visualization technology into teaching adds significant overhead to workload (Naps, et al., 2003). Based on the difficulties experienced by educators and using knowledge on active and adaptive learning, we have developed an approach based on the principles of reuse, customization and minimized coupling (Boyle, 2003). In his work Boyle outlines a framework for creating reusable and "re-purposable" learning objects (*ibid*, p.1) using principles from software engineering and pedagogy.

3. AN APPROACH TO DEVELOPING VISUALIZATIONS

This paper focuses specifically on the teaching of concepts related to RAM. This aspect of learning programming is in the centre of this investigation because the literature suggests that students find topics related to RAM handling during program execution particularly difficult (Milne & Rowe, 2004). This was also the experience of the authors of this paper. It was a constant struggle semester after semester: how to explain memory allocation for variables, how to illustrate an assignment statement not being the same as equality, how to show the difference between passing parameters by value and by reference. Both teaching experience and previous research suggest that graphical illustration can help understanding (Card, et al., 1999; Hundhausen & Douglas, 2000).

Here we present a study based on interpretive case study research. Research investigations focusing on understanding dynamics within a particular setting often employ case study as a research strategy (Yin, 1994). In this study we examined the teaching of programming concepts to first year students using the Visual Basic environment as a tool for concepts teaching and learning. Unfortunately as is the case in most universities (Prasad & Fielden, 2003), the subject was timetabled as teacher-centered lectures and separate laboratory classes where students were engaged in practical learning activities. However, due to small enrolment numbers (20-40 students), it was possible to introduce a large interactive component into the lectures, engaging students in exercises and discussions of programming concepts.

In the past the authors' classroom approach was to draw RAM on a whiteboard as a collection of cells and use different colored markers to show changes during program execution. But it was always messy, and often students while trying to listen and make notes would lose track of which instruction execution was being explained. Because of the ephemeral nature of the drawings this approach was also of little help for students who missed the class or who did not remember the explanation and needed to view it again.

To demonstrate dynamic changes in program execution these examples were embedded into lectures as animated PowerPoint presentations. A separate slide was devoted to each statement in the code (or even a specific part of the line in the code which was emphasized in the explanation). That line of code was enlarged to stand out. At the same time an animation was showing what happens when this bit of code executes with narration providing a verbal explanation of what the eyes see. For example, the slide captured on **Error! Reference source not found.**(a) demonstrates the creation of value parameter Y (i.e. a copy of the actual parameter). In the figure memory locations used by the procedure Button1_Click() are depicted in red and memory locations used by the procedure calculateSum() are in blue. The intention of color coding is to help students better understand memory allocation and use during program execution. The figure depicts the enlarged actual parameter Y in the procedure call and the enlarged corresponding formal parameter from the procedure header - "By Val Y As Integer". At the same time in the RAM area a blue box labeled Y

appears and PowerPoint shows an arrow from the red box Y to the blue box Y, meaning that at run time the value of 40 from the red box will be copied to the blue box. The color change effect is used to show that red Y with the value of 40 is controlled by the procedure Button1_Click() whereas the blue box Y belongs to the procedure calculateSum(). When 40 moves from the red box to the blue box it gradually changes its color from red to blue.

In response to the positive feedback from students on these presentations multiple examples of small animated program excerpts illustrating a programming concept were produced. Using one presentation as a template it was easy to develop multiple similar presentations showing simple programs with small modifications (e.g. by changing values and/or variable names).

Over time a library of such presentations was accumulated. However, while this approach helped students attending lectures it did not solve the problem of concepts revision or help students who missed the class. However modern technology and software developments can now easily solve these problems. There are user friendly screen capturing packages with voice recording (for example Adobe Captivate, Camtasia, BB Flashback). Using these packages we developed videos containing animated explanations of concepts with narration. These videos were produced as Flash animations for viewing over the Web as well as avi and exe files that students could run on their computers without depending on the Internet and website availability.

In the process of developing the videos we noticed that certain explanation statements are repeated again and again. Some slides were also repeated from one presentation to another and the explanation of those slides had to be narrated in exactly the same way again and again. It made more sense to record explanations as short mp3 files and attach them to slides.

Since Australia is a multicultural country and many academics are migrants speaking English with various accents, accents sometimes interfere with student understanding of the explanations. Sometimes the difficulty is not in understanding the material as in understanding what the instructor is saying. Since narrations in this project were recorded in mp3 format, we explored text to speech software utilization in replacement of self-recording. (Conversion facilities are available as standalone software such as Natural Reader or through the Web, e.g. http://vozme.com or http://spokentext.net).

The multimedia library for teaching programming is now comprised of PowerPoint slides and reusable audio files. The slides can be easily modified to show program excerpts in another programming language whereas the same narration explaining concepts are still applicable. Multiple building blocks that can be mixed and matched to create new learning materials have been collected. These building blocks are being developed in accordance with the principles of reuse, low coupling and easy customization as suggested by Boyle (2003).

To evaluate students' views on the use of visualizations in lectures and workshops, and for revision, we asked them to provide their opinions in a feedback questionnaire (the full questionnaire as well as the summary of responses are presented in the Appendix A). 28 students in the class replied to this questionnaire. On the question "Which of the two methods was more helpful in understanding the concepts of parameter passing? a) diagrams on the whiteboard; b) PowerPoint animations" 2 student selected (a), 6 students selected both and 20 students selected (b). On the second question "Would PowerPoint animations with recorded narration be sufficient for you to understand the concepts without attending the class?" 24 students replied "yes" and only 4 said "No".

Past research has shown that passively watching videos is of lower educational value than active learning engagement (Naps, et al., 2003) even though video materials are a valuable supplement to lectures and revision sessions. Consequently, several approaches to more active ways of engaging students were explored.

One such approach engaged students constructing their own visualizations as recommended by Hundhausen and Douglas (2000). Building on this idea we approached some of the students who responded to the questionnaire and asked them to develop animations illustrating code segments that were used as exercises for manual tracing in class. As the next step students were asked to produce their own code examples and develop animations in PowerPoint illustrating changes in RAM as their code executes. Their feedback acknowledged the benefit of such additional activities.

The activities were extended further with past programming students who were using skills learnt previously and applying them in the last year subjects developing software for real clients. Information Systems courses often have multimedia subjects or computer project subjects where students are learning to work with real clients. The authors signed up as clients for one of these subjects and ex-students developed some of the materials to support future teaching. Students involved in this activity were debriefed at the end and their feedback indicated that they valued this work because in addition to learning about real workplace environment they got the opportunity to revise the programming concepts in the context of a real project.

Benefits

Every instructor has their own way of presenting material. Having access to such building blocks of teaching materials makes it easy to choose the most applicable for the concept to be explained or even to easily replace any block with an improved version.

Creating the described materials does not create a significant additional workload for instructors.

Students can be involved in the learning materials development and therefore get additional opportunities for their own learning.

The described approach illustrates a way of producing multiple additional learning modules with a gradual increase of difficulty levels and tailoring them to individual students' needs as recommended in adaptive learning frameworks. For example, for fast learners 3-4 modules explaining parameter passing could be sufficient, whereas for slower learning students 6 or more modules could be assembled to gradually take them from the concepts of passing parameters by value to passing by reference. Appendix B presents a sample of six exercises that could be used for animations and code tracing. These exercises show gradual introduction of the parameter passing concepts from a business context to an abstract concept and from value parameters to the mix of value and reference parameters. For fast learning students the exercises 1, 4 and 6 could be sufficient for the understanding of these concepts. However, slower learning students may need a more gradual increase in the difficulty level, which is easy to achieve by applying small modifications to the existing slides and developing audio narration by adjusting pre-existing textual explanations and converting them to audio. When the multimedia library has a sufficient number of examples, it is possible for the instructor to suggest a longer set of exercises to a student or it can be left to a student to control the learning speed. It is also possible to give exercises to students for manual tracing and provide animated solutions for verification of answers.

4. CONCLUSIONS

This paper describes an approach for using readily available technologies and software to produce teaching and learning materials as small building blocks. The proposed and tested approach was implemented for teaching programming concepts within the frameworks supporting active and adaptive learning environments. The techniques discussed make it easier for instructors to develop customized learning modules, adapted to meet the individual needs of their students. This approach provides opportunities to engage students both in learning activities using the developed materials and to involve them actively in the material development process. Future work will harness the capabilities of interactive materials and the software supporting their development to even further promote active learning.

5. REFERENCES

Bennedsen, J., E.Caspersen, M., & Kölling, M. (Eds.). (2008). Reflections on the Teaching of Programming. Methods and Implementations (Vol. 4821): Springer-Verlag.

Bonwell, C. C., & Eison, J. A. (1991). *Active Learning: Creating Excitement in the Classroom*. Washington, DC, USA: George Washington University.

Boyle, T. (2003). Design Principles for Authoring Dynamic, Reusable Learning Objects. *Australian Journal of Educational Technology, 19*(1), 46-58.

Bruce-Lockhart, M. P., & Norvell, T. S. (2000). *Lifting the Hood of the Computer: Program Animation with the Teaching Machine.* Paper presented at the Canadian Conference on Electrical and Computer Engineering, Halifax, NS , Canada.

Card, S. K., Mackinlay, J., & Shneiderman, B. (1999). Readings in Information Visualiza-

tion: Using Vision to Think. San Francisco, California, USA: Morgan Kaufmann.

Donmez, O., & Inceoglu, M. M. (2008). *A Web Based Tool for Novice Programmers: Interaction in Use.* Paper presented at the the 8th International Conference on Computational Science and Its Applications (ICCSA 2008), Perugia, Italy.

Dufresne, R. J., Gerace, W. J., Leonard, W. J., Mestre, J. P., & Wenk, L. (1996). Classtalk: A Classroom Communication System for Active Learning. *Journal of Computing in Higher Education, 7*(2), 3-47.

Gomes, A., & Mendes, A. J. (2007). *Learning to Program - Difficulties and Solutions.* Paper presented at the International Conference on Engineering Education - ICEE 2007, Coimbra, Portugal.

Hadjerrouit, S. (2008). Towards a Blended Learning Model for Teaching and Learning Computer Programming : A Case Study. *Informatics in Education, 7*(2), 181-210.

Hundhausen, C. D., & Douglas, S. (2000). *Visualizations to Learn Algorithms: Should Students Construct Their Own, or View an Expert's?* Paper presented at the IEEE Symposium on Visual Languages, Los Alamitos, California.

Jenkins, T. (2001). *Teaching Programming - A Journey from Teacher to Motivator.* Paper presented at the 2nd Annual Conference of LTSN Centre for Information and Computer Science, Loughborough, UK.

Jenkins, T. (2002). *On the Difficulty of Learning to Program.* Paper presented at the 3rd Annual Conference of LTSN Centre for Information and Computer Science, Loughborough, UK.

Magoulas, G. D., Papanikolaou, K., & Grigoriadou, M. (2003). Adaptive Web-Based Learning: Accommodating Individual Differences through System's Adaptation. *British Journal of Educational Technology, 34*(4), 511-527.

Matthews, R., Hin, H. S., & Choo, K. A. (2009). *Multimedia Learning Object to Build Cognitive Understanding in Learning Introductory Programming.* Paper presented at the 7th International Conference on Advances in Mobile Computing and Multimedia (MoMM '09).

Milne, I., & Rowe, G. (2004). OGRE: Three-Dimensional Program Visualization for Novice Programmers. *Education and Information Technologies, 9*(3), 219–237.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., et al. (2003). Exploring the Role of Visualization and Engagement in Computer Science Education. *ACM SIGCSE Bulletin, 35*(2), 131-152.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., et al. (2007). A Survey of Literature on the Teaching of Introductory Programming. *ACM SIGCSE Bulletin 39*(4), 204-223.

Prasad, C., & Fielden, K. (2003). Introducing Programming: A Balanced Approach. *New Zealand Journal of Applied Computing and Information Technology, 7*(1), 89-94.

Rowe, G., & Thorburn, G. (2000). VINCE - an On-line Tutorial Tool for Teaching Introductory Programming. *British Journal of Educational Technology, 31*(4), 359-369.

Rudder, A., Bernard, M., & Mohammed, S. (2007). *Teaching Programming Using Visualization.* Paper presented at the 6th IASTED International Conference on Web-Based Education (WBE'07), Chamonix, France.

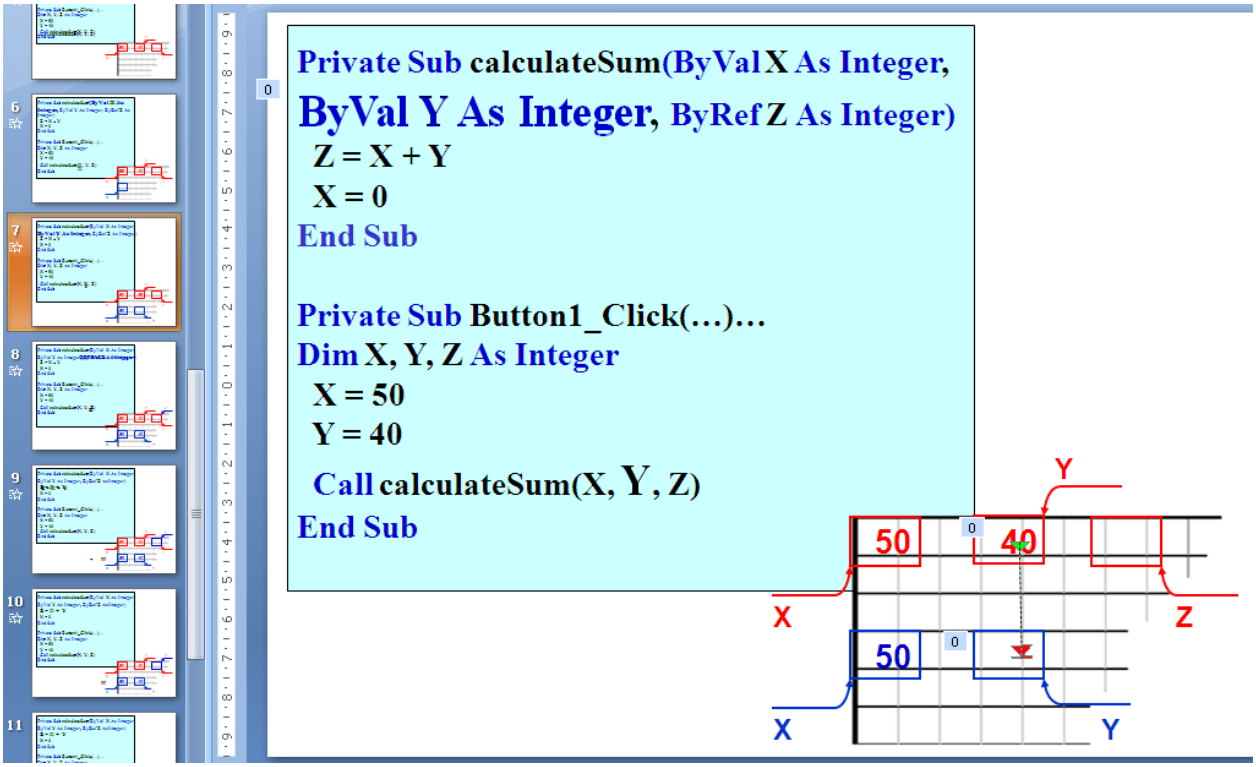
Satratzemi, M., Dagdilelis, V., & Evagelidis, G. (2001). A System for Program Visualization and Problem-Solving Path Assessment of Novice Programmers. *ACM SIGCSE Bulletin, 33*(3), 137-140.

Smith, P. A., & Webb, G. I. (2000). The Efficacy of a Low-Level Program Visualization Tool for Teaching Programming Concepts to Novice C Programmers. *Journal of Educational Computing Research, 22*(2), 187-215.

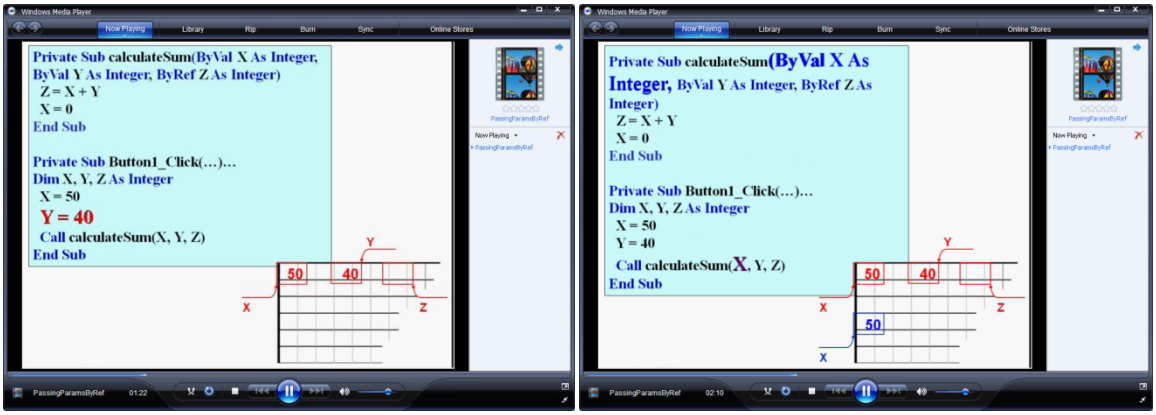
Spronken-Smith, R., & Harland, T. (2009). Learning to Teach with Problem-Based Learning. *Active Learning in Higher Education, 10*(2), 138-153.

Stone, J., & Clark, T. (2011). *The Impact of Problem-Oriented Animated Learning Modules in a CS1-Style Course.* Paper presented at the ACM Special Interest Group on Computer Science Education Conference (SIGCSE 2011).

Yin, R. K. (1994). Case Study Research. Design and Methods (Second ed. Vol. 5). Thousand Oaks, CA: Sage Publications, Inc.



(a) Screenshot of one of the PowerPoint animations: the slide shows creation of the value parameter Y in RAM



(b) Screenshots of the playing avi file

Figure 1 Animations on passing parameters

Appendix A. The questionnaire distributed to students

Dear student,

The following questionnaire is confidential, anonymous and participation is voluntary.

For explanation of differences between value and reference parameters you were shown examples similar to the one below:

```
Private Sub calculateSum(ByVal X As Integer,_ ByVal Y As Integer, ByRef Z As Integer)
  Z = X + Y
  X = 0
end sub

Private Sub Button1_Click()…
Dim X, Y, Z As integer
  X = 50
  Y = 40
  Call calculateSum(X, Y, Z)
end sub
```

Solutions were explained in two ways: by drawing diagrams on the whiteboard and by using PowerPoint animations with the lecturer narration. Please give us your feedback on the suitability of each of the methods:

1. Which of the two methods was more helpful in understanding the concepts of parameter passing:

    a. diagrams on the whiteboard
    b. PowerPoint animations

2. Would PowerPoint animations with recorded narration be sufficient for you to understand the concepts without attending the class?

    Yes / No

3. Would you use PowerPoint animations for revision in your own time as addition to class sessions?

    Yes / No

4. If you have any suggestions regarding animations, please use the space below to share your thoughts with us.

Summary of students replies to the questionnaire:

Question 1:
Chose “a” – 2 students, “b” – 20 students, wrote “both” – 6 students

Question 2:
Answered “yes” – 24 students, “no” – 4 students

Question 3:
Answered “yes” – 25 students, “no” – 3 students

Question 4:
9 students put additional comments. Most of the comments were stating that animations were a big help in understanding the concepts. 2 students mentioned that it would be good to see previous state of RAM as well. 1 student commented that whiteboard drawings are sufficient if different color markers are used. 1 student stated that whiteboard drawing and animations were equally good for understanding of the concepts.

Appendix B. Sample exercises on parameter passing

Example 1. Passing parameters by a value in a business context

```
Private Function calcArea(ByVal w As Integer,
ByVal h As Integer) As Integer
  return w * h
End Function

Private Sub btnCalculate_Click(...)…
Dim width, height, area As Integer
  width=3
  height=4
  area = calcArea(width, height)
End Sub
```

Example 2. Passing parameters by value in an abstract context

```
Private Function calcSum(ByVal p As Integer,
ByVal q As Integer, ByVal r As Integer) As Integer
  return p + q + r
End Function

Private Sub btnCalculate_Click(...)…
Dim num1, num2, num3, num4 As Integer
  num1=5
  num3=7
  num4=9
  num2 = calcSum(num4, num1, num3)
End Sub
```

Example 3. Passing parameters by value in an abstract context

```
Private Sub makeChanges(ByVal A As Integer,
ByVal B As Integer)
  B = 4
  A = 5
end Sub

Private Sub Button1_Click()…
Dim X, Y As integer
  X = 10
  Y = 20
  Call makeChanges(Y, X)
End Sub
```

Example 4. Passing parameters by value and by reference in an abstract context

```
Private Sub makeChanges(ByVal A As Integer,
ByRef B As Integer)
  B = 4
  A = 5
end Sub

Private Sub Button1_Click()…
Dim X, Y As integer
  X = 10
  Y = 20
  Call makeChanges(Y, X)
End Sub
```

Example 5. Passing parameters by value and by reference in an abstract context

```
Private Sub mixUp(ByVal X As Integer, ByRef Y
As Integer, ByRef Z As Integer)
  Z = X + Y
  X = 0
End Sub

Private Sub Button1_Click(…)…
Dim X, Y, Z As Integer
  X = 2
  Y = 8
  Call mixUp(X, Y, Z)
End Sub
```

Example 6. Passing parameters by value and by reference in an abstract context

```
Private Sub mixUp(ByVal X As Integer, ByRef Y
As Integer, ByRef Z As Integer)
  Z = X + Y
  X = 0
End Sub

Private Sub Button1_Click(…)…
Dim X, Y As Integer
  X = 50
  Y = 40
  Call mixUp(X, Y, Y)
End Sub
```