Abstract

Based on Mayer's (2001) model for more effective learning by exploiting the brain's dual sensory channels for information processing, this research investigates the effectiveness of using aural instructions together with visualisation in teaching the difficult concepts of data structures to novice computer science students. A small number of previous studies have examined the use of audio and visualisation in teaching and learning environments but none has explored the integration of both technologies in teaching data structures programming to reduce the cognitive load on learners' working memory.

A prototype learning tool, known as the Data Structure Learning (DSL) tool, was developed and used first in a short mini study that showed that, used together with visualisations of algorithms, aural instructions produced faster student response times than did textual instructions. This result suggested that the additional use of the auditory sensory channel did indeed reduce the cognitive load.

The tool was then used in a second, longitudinal, study over two academic terms in which students studying the Data Structures module were offered the opportunity to use the DSL approach with either aural or textual instructions. Their use of the approach was recorded by the DSL system and feedback was invited at the end of every visualisation task.

The collected data showed that the tool was used extensively by the students. A comparison of the students' DSL use with their end-of-year assessment marks revealed that academically weaker students had tended to use the tool most. This suggests that less able students are keen to use any useful and available instrument to aid their understanding, especially of difficult concepts.

Both the quantitative data provided by the automatic recording of DSL use and an end-of-study questionnaire showed appreciation by students of the help the tool had provided and enthusiasm for its future use and development. These findings were supported by qualitative data provided by student written feedback at the end of each task, by interviews at the end of the experiment and by interest from the lecturer in integrating use of the tool with the teaching of the module. A variety of suggestions are made for further work and development of the DSL tool. Further research using a control group and/or pre and post tests would be particularly useful.

The Effectiveness of Aural Instructions with Visualisations in E-Learning Environments

Fuad Hamad Alhosban

PhD Thesis

Technology Enhanced Learning Research Group

School of Engineering and Computing Sciences

Durham University

2011

List of Contents

List of Figures

List of Tables

Declaration

No part of the material provided has previously been submitted by the author for a higher degree in Durham University or in any other University. All the work presented here is the sole work of the author and no-one else.

Copyright

Acknowledgements

I would like to thank everyone who has helped me to complete this thesis. Without them, the work would never have been accomplished.

I am especially indebted to the following people.

My father, Hamad Alhosban who has provided continuous moral and financial support over the whole period of my post graduate studies. My mother's prayers and support gave me hope and motivation to continue through difficult periods.

My lovely wife, Samantha, gave me her love, patience and continuous support through both the easy and the difficult times of my research work.

Professor Elizabeth Burd provided me with brilliant supervision and supported me with ideas and advice that helped me greatly.

Dr Andrew Hatch has been a good friend, as well as an excellent second supervisor, and I have valued his advice on both personal and academic issues.

Dr Iyad Alagha has been a good and special friend who was always available to help, listen and give advice.

Finally, I record my respect and thanks to everyone in my family and to all my friends for their support and encouragement and for being part of my life.

Chapter 1 : Introduction

1.1 Research Overview

One of the first challenges that faces novice computer science (CS) students when they start their course is to acquire the skills required to write or compile computer programmes. Consequently, the Introduction to Programming and the Data Structures (PDS) modules are compulsory for first year CS students at Durham University, UK. Within these modules, object-oriented programming (OOP) is a widely used paradigm for software development.

OOP is defined by Snyder (1986, p1) as "a practical methodology that encourages modular design and software reuse." data structures, on the other hand, offer memory based organisation of information for better algorithm efficiency (McAllister, 2008, p.5). Understanding the concepts of OOP and data structures is crucial because they enable students to reuse existing code and to create objects that form the building blocks of their programming projects.

The starting point towards students acquiring professional programmer skills is to make sure that they participate in high quality learning. Students need to interact with their learning environment by talking, listening, reading, writing and reflecting on their own knowledge as they approach the course content (Meyers, 1993). This interaction helps students to be actively engaged with their learning environment. Active learning theories believe teachers and students to be actively engaged in their learning environment if they are exploring, experiencing, experimenting, testing and applying the knowledge they gain in class to solve real life problems (Longmire, 2000).

Visualisation is one of many attempts to use technology to improve learning by creating a mental image of how things work. It is also a common learning style that many students prefer as a way to increase their comprehension of concepts, bearing out the proverb “a picture is worth a thousand words.” As an active learning approach, interactive visualisation tools increase the interaction between the learner and the subject being studied (Evans and Gibbons, 2007).

The visualisation of algorithms is used to enhance learners’ experience and facilitate the understanding of the algorithm and its concepts. However, there are problems with many of the existing visualisations. These include loss of focus if the level of abstract representation focuses concentration on low-level steps rather than on high-level properties like invariants (Mudner and Shakshuki, 2004).

The use of audio, either musical or spoken sound, was presented by researchers (Gaver et al., 1991; Brown, 1992; Stifelman, 1995; Franklin, 2001; Vickers et al., 2005) as a means of aiding visualisation in learning environments. Its usage in CS learning and in algorithm animation started as a way to describe what visualisations are currently showing (Brown, 1992). It can also help students to focus on their learning task. This means that the use of both aural instructions and visual components can improve students’ awareness of their learning environment.

Cognitive approaches to human methods of learning, on the other hand, have highlighted the transformation that occurs on different mental representations of situations and tasks. To understand human cognition, some knowledge of models of human memory organisation is important. Kahneman (1973, p.173) and Navon (1984) explained that the concept of mental load was based on the concept of a communication between two channels with limited capacity. The Dual Channel Assumption (Mayer, 2001) believes that humans process information in two separate channels, visually or aurally. Research shows that the use of animation and an associated aural narration were most effective when presented simultaneously rather than successively (Kalyuga, 2006). In short, to provide students with an effective and active learning environment, the cognitive load should be reduced to the minimum.

This research investigates the use of focused visualisation with aural instructions in an interactive learning environment to help students to learn OOP and data structures. By exploiting the dual-channel approach, this integration is expected to reduce cognitive load on learners' working memory.

The research presented in this thesis was conducted at Durham University. It involved first year CS students in the School of Engineering and Computing Sciences who undertook both the IP and the PDS modules. These modules introduce the concepts of OOP and data structures as a starting point in learning programming using the Java programming language. However, the approach used in this research may be applied to learning any other programming language.

1.2 Definition of Terms

To establish a solid theoretical structure for this thesis, a set of frequently used key terms are listed and defined below:

- Aural Instructions

Interactive high quality computer generated speech in the form of short instruction that helps students to complete a learning task.

- Cognitive load

The amount of mental effort needed by the brain to learn something, solve a problem or correctly respond to instructions.

- Data Structures

The concept of dividing programming problems into smaller pieces and creating building blocks that are called objects. This research refers to two types of data structures: Linked Lists and Binary Trees.

- Data Structure Learning (DSL) Environment

An environment designed as a centre for CS students' learning activities; it provides the content and resources required to help make the activities successful. It targets students who are studying the Programming and Data Structures modules. The DSL environment consists of the DSL approach and the DSL tool.

- DSL Approach

The approach adopted by this research. It is an interactive learning method using visualisation with aural instruction to teach CS students the concepts of data structures programming.

- DSL tool

A learning tool that implements the DSL approach through an interactive software application that helps CS students to learn the concepts of data structures.

- E-Learning

Delivery of a learning, training or education programme by electronic means, such as a computer or electronic device (e.g. a mobile phone).

- Multimedia Learning

A method for simultaneously presenting visual and aural content to provide training, educational or learning material.

- Practical Session

A compulsory two hour class where students practice programming lessons and solve their programming assignments. These sessions take place at the School of Engineering and Computer Sciences labs in Durham University.

- Textual Instructions

Interactive instructions shown on screen in the form of written text that helps students to complete a learning task.

- Visualisation Task

A sub program of the DSL tool which enables students to use the interactive visualisation to learn the concepts of data structures. The DSL tool contains four visualisation tasks: Learning About Objects, Linked Lists, Binary Search Tree, and Binary Tree Traversal.

1.3 Research Objectives

Previous research (Kazi, 2000; Cross, 2004; Karavirta, 2004; Bednarik, 2006; Culwin, 2006; Rajala, 2008; Wolf, 2008; Briana, 2009) in this field examined the use of visualisation tools to assist students' learning experience. This research investigated the reduction of cognitive load by providing an intensive visualisation environment. However, not all visualisation environments reduce the cognitive load, and Tudoreanu (2003, p106) argues that a “visualization environment that requires users to handle additional information and tasks, which increases cognitive load, offers similar performance advantages to that of a user who has no visualization at all.”

Other research (e.g., Brown, 1992; Franklin, 2001; Vickers, 2005; Qiu and Benbasat, 2005) investigated the use of sounds or audio feedback to improve the interaction between users and their systems. However, their investigation did not exploit the dual channels of working memory to speed the learning process.

The main goal of this thesis is to investigate the effectiveness of using aural instruction together with visualisation in teaching the concepts of data structures to novice CS students.

1.4 Research Contributions

The most important contributions of this research are:

- Development of a learning environment and a learning approach that facilitates the use of both aural instructions and visualisation to assist students in learning the concepts of data structures, and a tool that implements the approach called the Data Structures Learning (DSL) tool.
- Obtaining quantitative data to investigate the ways students interact with a new learning method that uses visual and aural instructions when learning the concepts of data structures.
- Obtaining qualitative data that measures the effectiveness of using the DSL tool and its impact on the students’ learning experience.

- Using an active learning approach to assist CS students' learning based on interacting with the DSL tool.

In addition, the thesis will answer a set of research questions shown in Table 1.1.

1.5 Criteria for Success

In order to reach conclusions about the learning issues raised in Section 1.3, it is important to propose and answer a set of research questions. These are approached by proposing four hypotheses concerned with cognitive load, student perception, and outcome. The success of this research will be measured by the clarity of its answers to the research questions and its conclusions about the hypotheses.

Table 1.1 shows the four hypotheses and the nine research questions.

H1	**Reducing cognitive load improves student engagement and outcomes when learning data structures.**
Q1	What existing research evidence is there that the simultaneous use of aural instruction along with visualisation will reduce cognitive load when learning data structures?
Q2	Does the combination of aural instruction and visualisation reduce students' response time for task completion compared with textual instruction and visualisation?
H2	**The use of aural instructions in teaching data structures to CS students has a positive effect on student perception of data structure concepts.**
Q3	Do CS students perceive benefits from aural instructions along with visualisation when studying data structures?
Q4	Is there a relationship between visualisation type and CS students' choice of instruction type?

Q5	Do students prefer the DSL environment based on visualisation only regardless of the type of instruction?
H3	**Students perceive a positive benefit to their learning by using the DSL tool.**
Q6	Do CS students choose to use the DSL tool while studying data structures?
Q7	Of the three data structure types used in this study, which do CS students select to explore through the DSL tool?
Q8	Do CS students perceive benefits from using the DSL tool to build a mental model of data structures?
H4	**There is a positive relationship between a CS student's level of engagement with the DSL tool and his/her level of achievement, as measured by the official assessment marks.**
Q9	What is the level of achievement of CS students who choose to use the DSL tool the most?

Table 1.1: Research Hypotheses and Research Questions

There follows a justification of how the proposed research questions support investigating the research hypotheses.

1.5.1 H1: Reducing cognitive load improves student engagement and outcomes when learning data structures.

The first element that the research investigates is the reduction of cognitive load in learning environments. This hypothesis claims that reducing cognitive load will result in a better engagement of students with the learning environment and, thus, improve their learning outcomes.

Q1 focuses on finding evidence from previous research in the field of multimedia learning that the simultaneous use of aural instruction with visualisation will reduce cognitive load. The evidence should also show that the reduction of cognitive load leads to a reduction of students' response time to a set of instructions. Q2, on the other hand, investigates how students' response time to instructions is affected by the format of the instruction, e.g.: textual, aural or textual with aural.

1.5.2 H2: The use of aural instructions in teaching data structures to Computer Science students has a positive effect on student perception of data structure concepts.

The second element of the research concerns students' perception of the research approach. This research claims that the use of aural instructions with visualisation can help CS students in learning the concepts of data structures.

Q3 investigates whether or not students perceive benefits from the use of aural instructions with visualisations when they study the data structures types. Students' responses to a questionnaire can provide direct evidence about the benefits of using the DSL environment, and the extent to which the approach provided students with a positive effect on their perception of data structure concepts.

Q4 investigates if there is a relationship between the types of data structure visualised and students' choice of instruction format (aural or textual). The data generated during students' experience of the learning environment can show if the use of aural instructions had a positive effect on student perception of some data structure concepts, and whether or not they tended to use the aural instruction format in learning the concepts that are harder to understand. On the other hand, Q5 investigates if students preferred to use the interactive learning environment based on visualisation only, without any consideration of the format of instruction.

1.5.3 H3: Students perceive a positive benefit to their learning by using the DSL tool.

Also within the second element of the research, that is students' perception, this research hypothesises that there is a positive benefit of using the prototype tool created to test the research approach validity. Q6 investigates if students choose to use the DSL tool in practice while studying the proposed data structure. The frequency of student usage will help to determine whether or not the DSL tool benefits student learning. Q7 investigates which of the three visualised data structures engaged students the most. The answers to this question will provide information about the preferred visualisation task among all the students and show the level of positive benefit perceived from each visualisation task.

Q8 investigates if students perceive benefits from being able to build a mental model of the proposed data structures. The answer to this question can provide information about the overall student experience with the DSL approach, and whether or not it helped them in meeting their expectations of the DSL tool.

1.5.4 H4: There is a positive relationship between a CS student's level of engagement with the DSL tool and his/her level of achievement, as measured by the official assessment marks.

The third element of the research investigates the outcomes for students of using the designed learning environment. The CS students' level of engagement with the interactive learning environment will be compared with their official end-of-year assessment marks. Q9 investigates whether or not students' level of achievement has a relationship with the duration of their engagement with the DSL tool. The results will provide information about what type of students are keen to use the DSL tool.

1.6 Thesis Outline

This is the first of seven chapters in this thesis. The others are outlined below.

Chapter 2 provides an overview of the research literature related to collaborative teaching tools that claim to enhance students' learning experiences. Such tools implement the use of software visualization, aural instructions and interactive learning environments. To cover the learning process from all aspects, this chapter also reviews students' perceptions, and their willingness to engage with new approaches.
Chapter 3 describes with the implementation of the tool that was developed to evaluate the approach to learning of this research. However, it starts by introducing the adopted DSL learning environment and discussing the reasoning behind adopting the DSL approach and how it relates to the research questions. The discussion of the implementation of the DSL tool focuses on some of the technical details of the tool's components and functionality. It describes the methodology of visualising interactive DS concepts in easy-to-use sub-programs called tasks, as well as integrating Text to Speech (TTS) technology to provide spoken feedback to students.

Chapter 4 focuses upon the pilot study designed to test the functionality and reliability of the research environment, and to answer research question Q2. This ensures that the correct path is set to answer the rest of research questions. The study investigates the use of the DSL environment to assess the effectiveness of using textual and aural instructions in a visualised CS learning environment by measuring students' response time to aural and textual instructions.

Chapter 5 discusses the methodologies employed in this research to answer the research questions. The first part will look at students' learning objectives and the features of a learning tool that will match students' requirements with the least possible cognitive load on the students. It will describe the technical information involved in developing the DSL tool. Finally, it will describe the experimental design of the research.

Chapter 6 discusses the evaluation of the DSL environment and provides an overview of the students' engagement with the DSL tool during the course of the study. It details the results of the data collected in this research by both quantitative and qualitative methods.

Chapter 7 analyses the overall results of using the DSL environment, and evaluates the quantitative and qualitative data obtained. The analysis of the results will answer the research questions listed in Table 1.1 and decide if each of the research hypotheses have been proved or disproved. It also presents the conclusions of this study and considers the future work that could be developed as a result of this research.

Chapter 2 : Literature Review

2.1 Introduction

Understanding the concepts of Object Oriented Programming (OOP) and data structures is crucial for CS students. OOP and data structures are important because they enable the reuse of existing code and the creation of objects that form the building blocks of a student's project. In turn, these shorten the time needed by programmers to provide code in a wide range of academic exercises, and in the real world. However, this will be hard for student programmers to achieve if they fail to understand the underlying theoretical assumptions.

The starting point towards developing professional programmer skills is to make sure that the students have meaningful learning and that they are able to adapt to new technologies in different circumstances. The quality of learning, however, depends on the curriculum and in-class activities. It also depends on the approach adopted by teachers to deliver the knowledge skills required. Thus, the point is to provide students with activities that open their eyes to the new world of creativity in programming techniques and well-structured ways of thinking.

In order to help students to understand the concepts of data structures, it is necessary to look at the way each student learns. Adapting to students' learning styles requires that teachers deliver learning materials that suit every student. Different approaches will be discussed in this research. To adapt to students' learning styles and thus enhance the learning process, a collaborative environment needs to be created. Furthermore, teaching towards each of the main learning styles needs to be considered in order to provide a suitable learning environment for each learner.

Using visualisations and algorithm animation in the classroom may support one of the student learning styles. Many researchers have developed visualisations to enhance students' learning experience and open student minds to engage with the concepts rather than memorising them as facts or chunks of knowledge (Brown, 1991; Shilling and Stasco, 1992; Culwin, Adeboye and Campbell, 2006). This enhancement of learning experience is the main purpose of attempting to change from the passive reception of information to involving students more in the learning process.

Aural based interfaces can also enhance visualisation tools and thereby support another main leaning style. This research will look at studies on supporting visual interfaces with auditory instruments and examine research on associated issues, such as the use of audio and visualisation together to help people with visual disabilities or to create another way of communicating with students who learn more by listening.

This research will cover the interactive and collaborative technologies that engage students in the learning process and give them the chance to create, update and even interact with their learning materials.

2.2 Learning and Teaching

2.2.1 Active and Passive learning

Active learning requires a learning environment which allows students to interact by talking, listening, reading, writing and reflecting on their own knowledge as they deal with the course content. They can do this through problem-solving exercises, informal small groups, simulations, case studies, role playing, and other activities, that require students to apply what they are learning (Meyers, 1993). Active learning is an attitude towards learning, and a teaching method, that encourages the student to play a more active role in his or her learning. However, some people (e.g., Roberts, 2001) have argued that listening to lectures, watching films or television or browsing the web are activities that do not require student engagement and that encourage learners to be passive.

What is really required is that students should be engaged in class of higher-order tasks so that they do more than just listening. Such tasks involve analysis, evaluation and synthesis. Active learning approaches have been introduced by CS educators into their classrooms with students working alone or in groups solving problems or participating in activities during lectures. Individual solutions can be shared with the class, for example, by asking students to write their answer on the board or overhead projector. This is a classical interactive exercise by which an individual student's solution benefits the whole class. All students are helped by seeing other students making mistakes similar to their own or by seeing more advanced students present a particularly elegant solution (Simon et al., 2004). In addition, such interaction gives the instructors a clear idea about the students' level of understanding and helps them to decide on the best way to teach the subject.

In contrast, passive learning takes place in a more traditional classroom where instructors verbalise the information and student activity is confined to taking notes. Students are assumed to enter the course with minds like empty vessels or sponges to be filled with information without necessarily considering the theory behind it. Some instructors consider this method is based on common sense (McManus, 2001).

McKinney (2007) argues that students who prefer passive learning or students in large classes may show some resistance to active learning, especially if they are familiar with traditional lectures. Therefore, there is a need to prepare students for active learning methods by explaining to them the objectives and benefits of such methods. Some appropriate training for the teachers may also be needed to help to implement active learning techniques (Niemi, 2002).

Active learning can be implemented in or outside the class. The implementation of educational tools can be through computer simulations, internships, online assignments, Internet discussion lists, or independent study (McKinney, 2007). Active learning can be used with all levels of students from first year undergraduates through to graduate students. Teaching a full class does not prohibit the use of active learning techniques; in fact, in a large class, they may be especially important to promote interest and learning (McKinney, 2007).

2.2.2 Deep and surface learning

The concepts of deep and surface learning are the two key approaches to the way we learn and interact with the surrounding environments. Marton and Saljo (1976) carried out the original work on approaches to learning. In an experiment, two groups of students were given an academic text to study, and they expected to be questioned about this text. Each group adopted a different approach. One focused on memorizing some facts that they expected to be quizzed about, while the other tried to understand the whole meaning of the text. The first group can be called surface or superficial learners while the other can be considered as deep learners. Atherton (2005, para.2) identified a third approach called "achieving" learning, defined as "a very well-organised form of surface approach, and in which the motivation is to get good marks."

Deep learners concentrate on what is important and they relate what they are learning to their previous knowledge. Deep learners tend to implement the acquired knowledge in their daily lives. They also organise and structure content into a coherent whole, and this all comes from the learners' own efforts. On the contrary, surface learners tend not to relate problems they encounter to a main concept, and tend not to differentiate between principles and examples, as their focus is how they will be assessed (Ramsden, Beswick and Bowden, 1989).

Deep approaches to learning in higher education have been intensively researched (e.g., Atherton, 2005; Notess, 2006; Smith, 2007) and they are now widely encouraged as an optimal way of module content delivery, although deep learning is not easy to achieve. In order to push students more towards deep learning, Notess and Neal (2006) suggested five approaches that an instructor needs to apply:

- Make sure that the course is well organized, paced, and communicated. Otherwise, students will become disappointed, discouraged or frustrated.
- Develop activities that are authentic and feel more real than imitation, and relate them to the student by letting real world constraints play a part.
- Give the students more control of the course content by allowing them to select the required reading or the type and topics of their assignments.

- Select activities that cannot be completed without application, analysis, synthesis and evaluation, thus challenging the students and raising the standard of discourse.

An important constraint on the promotion of deep learning outcomes among students is that their teachers might not have been given enough training, tools and time to engage in practices that contribute to these outcomes (Smith and Colby, 2007, p:205). Teachers must promote intentional rather than accidental efforts to enable deep student learning.

2.2.3 Interactive learning

Allen (1999) describes interactive learning as a process that enables students to obtain information by combining traditional resources, such as textbooks, with hands-on activities. In interactive learning, students work together in groups or they interact with tutoring software or any appropriate media tool. Interactive learning is also learning by doing and experimenting with knowledge in order to understand it. It is considered an important learning style.

In order to provide interactive learning and achieve deep learning, instructors need to use the latest online technologies and adopt the new communication mediums that students are already using (Rajasingham, 2010). Even before students start their higher education, technologies that enhance collaboration between staff and students, and between students themselves, can be used.

2.2.4 Constructive alignment

Biggs (2003) created the term ‘constructive alignment’ to express the underpinning concept behind the requirements for programme specification, declarations of Intended Learning Outcomes and assessment criteria, and the use of criteria based assessment (Houghton, 2004). Biggs’ notions support active learning by encouraging learners to construct their own knowledge rather than adopting passive learning and accepting whatever they are told. Biggs (2003, p.13) summarises his approach in his claim that “Education is about conceptual change, not just the acquisition of information."

Alignment requires teachers to set up learning environments that support the learning activities appropriate to achieving the desired learning outcomes. The teaching methods should be aligned with the learning activities assumed in the intended outcomes, and this should be done in a way that engages the learner and ensures that the intended learning outcomes are achieved.

Biggs (2003) suggests four main steps to achieve constructive alignment:

1. Defining the intended learning outcomes (ILOs)
2. Choosing teaching/learning activities likely to lead to the ILOs
3. Assessing students' actual learning outcomes to see how well they match what was intended
4. Arriving at a final grade

It is argued that a well-designed course will hinge on a close relationship between these essential elements. A poorly designed course will not develop these close relationships and consequently it will be difficult for learners to achieve their desired learning goals.

This research presents the approach of using aural instructions combined with a visualised interactive environment. This environment is designed to match the intended learning outcomes of the Programming and the Data Structures modules designed for novice students studying Computer Science at Durham University. These learning outcomes will be discussed in detail in Section 3.2.1. Based on Biggs' constructive alignment theory, this thesis aims to apply the constructivism process. This process is met by making sure that the end product of the research meets the Computer Science Department's stated learning outcomes. It also introduces a new learning activity through producing visualisation of the concepts of data structures. This is based on creating an interactive tool that helps students in their learning process to meet their ILOs.

2.2.5 E-Learning

E-Learning or Electronic Learning has been defined in many different ways depending on the context (Stockley, 2004; Nycz, 2007). According to Stockley (2003, p.1), electronic learning is "The delivery of a learning, training or education program by electronic means. E-Learning involves the use of a computer or electronic device (e.g. a mobile phone) in some way to provide training, educational or learning material."

The term E-Learning is generally used to refer to the use of electronic tools in a learning process. However, there has been disagreement about the accuracy of this definition, as the use of some electronic devices, such as a microphone or a data projector, does not constitute E-Learning. Tavangarian (2004, p.2) suggests that E-Learning could be defined as "all forms of procedural electronic supported learning and teaching" that aims to "affect the construction of knowledge with reference to the learner's individual experience, practice and knowledge." E-Learning is the use of new technologies, applications and networking to improve the process of learning and help the learner to get the most knowledge from their studies.

Mayes (1999) argues that there are three fundamental stages of learning and these can be supported by three kinds of courseware. Technology based learning, he says, involves a cycle of conceptualisation, construction and dialogue, as shown in Figure 2.1. At the conceptualisation stage, learners review other people's learning resources online, such as lecture slides or other external information. In the construction stage, learners apply the knowledge they have acquired to meaningful tasks, and this can be done by using computer assisted assessment or online tests. Finally, at the dialogue stage, actual learning takes place when the learners get feedback from their instructors about their performance and this can be supported by using online discussion or any other form of online social interaction.

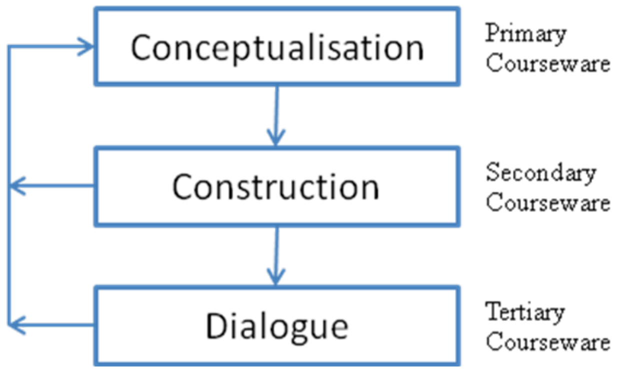


Figure 2.1: Mayes Learning Cycle

Salmon (2002, p.3) presented the concept of online learning activities “e-tivities” as frameworks to enhance active online learning by individuals or groups. The importance of e-tivities comes from their ability to produce useful pedagogies for learning by focusing on their implementation by means of network technologies. Although Salmon argues that e-tivities can make a key difference in learning, little enjoyable and cost effective online teaching has been produced.

2.2.6 E-Learning Technologies

There is a long history of failed attempts to improve the processes of teaching by using technological innovations like radio and TV (Warschauer and Healey, 1998; Salaberry, 2001). However, there is still some optimism that it can be done, especially if crucial parameters, or crucial characteristics, are satisfied. Nicolaou and Constantinou (2005) listed four such parameters:

- Networking technologies that provide the communication capabilities that are tuned to the requirements of teaching and learning.
- The learning sciences have a better grasp of the requirements offered by technological solutions.
- Research tools support the process of implementation through the evaluation of learning outcomes.
- The awareness by teachers of the need for improved teaching methods to achieve higher quality education.

The new technologies have proven E-Learning potency and the capability to reach all kinds of people, at different learning levels. This also applies to non-students who use the new technologies to learn something. Such technologies include screencasts, ePortfolios, Personal Digital Assistants (PDA's), MP3 Players with multimedia capabilities, web-based teaching materials, hypermedia in general, multimedia CD-ROMs, web sites and web 2.0 communities, collaborative software, e-mail, blogs, wiki, text chat, computer aided assistance, educational animation, simulations, games, learning management software, electronic voting systems, virtual classrooms and many others.

E-Learning has proved valuable beyond traditional education and can be exploited for the long term learning requirements of its users, such as for Collaborative Professional Development (CPD). This research is related to E-Learning because it proposes a prototype tool to be used in CS labs to provide interactive visualisation with aural instructions. This E-Learning technology aims to help students in their learning process.

2.2.7 Learning Styles

The term ‘learning styles’ refers to the ways in which students prefer to learn. They include seeing and hearing, reflecting and acting, reasoning logically and intuitively, and analysing and visualizing (Felder and Soloman, 1988).

Felder and Soloman (1988) categorised four main learning styles. Active and reflective learners tend to understand information by doing something active with it. Sensing and intuitive learners like to learn facts and they seek possible relationships between them. Visual and verbal learners get more information from seeing or from written or spoken words than from any other format. Finally, sequential and global learners depend on the order of the information presented to them.

2.2.7.1 Why Learning Styles?

The interest in students’ learning styles started in the mid-1980s. Felder and Soloman (1988) suggested changing teaching methods so that they suited students’ ways of learning. They did not call for a radical change in teaching methods, but rather for the systematic adoption of some instructional techniques to suit the variety of learning styles found in students.

Kolb (1984, p 21) also presented an early learning style model. This used terms such as experiential learning theory (ELT) and learning styles inventory (LSI).

Kolb’s model operates at two levels and it identifies four main types of learning: concrete experience, reflective observation, abstract conceptualization, and active experimentation, as illustrated in Figure 2.2. The second level distinguishes four main learning styles: diverging, assimilating, converging, and accommodating (Kolb, Boyatzis and Mainemelis, 2001, p 139). Kolb’s learning model proposes that there are four types for learning abilities, namely, Experiencing, Reflecting, Conceptualising, and Planning, and learners choose which of these they will use in different learning situations.

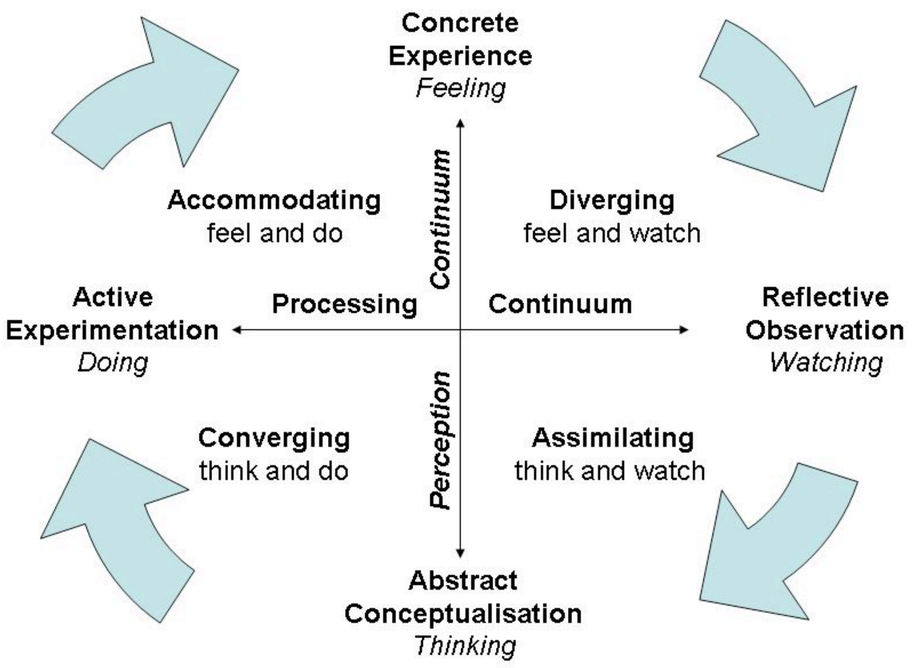

Concrete Experience *Feeling*
Diverging feel and watch
Reflective Observation *Watching*
Assimilating think and watch
Abstract Conceptualisation *Thinking*
Converging think and do
Active Experimentation *Doing*
Accommodating feel and do
Processing
Continuum
Perception
Continuum

Figure 2.2: Kolb's Learning Styles and Experiential Learning Model (Clark, 1995)

Rosati, Dean and Rodman (1988) studied the relationship between students' learning styles and instructors' teaching styles. In an experiment to explore the interaction between the learning styles of students and the way they were taught, two different presentations were made to two similar heterogeneous groups of engineering students. One was designed for sensing style students who rely on experience rather than theory and have a preference for advancing from their starting knowledge in a step-by-step manner. The other was designed for intuitive students who tend to rely on intuition and inspiration and who are often more able to understand abstract, symbolic and theoretical relationships. The Myer-Briggs type indicator (MBTI) of personality was used in this experiment as an indicator of the students' learning-style preferences. The results showed that students' performance can be improved if teachers recognize the importance of individual learning styles.

2.2.7.2 Computer Science and Learning Styles

Howard et al. (1996) looked at integrating Felder's (1988) learning styles, Klob's (1984) learning cycle and Bloom's (1956) taxonomy, which they described as "a hierarchical representation of the students depth of knowledge in a given subject or cognitive domain" (p.227). They developed a blueprint that used a number of teaching tools with one computer science class and, over an entire semester, divided the time equally to meet the requirements of each type of learner. They concluded that there are many techniques and tools available to improve students' performance in the classroom.

Dunn and Dunn's (1990) Learning Styles, also known as the Visual, Auditory and Kinaesthetic (VAK) learning model, has been widely used in schools in the United States. Its major components include the model's principles, its learning style elements, identifying each student's learning style, and its impact on the dimensions of the instructional situation. The main principle or theoretical assumption of this model is that "Most individuals can learn" (Dunn, 1990, p.1). Instructional environments, resources and approaches depend on diversified learning style strengths. Everyone has strengths, but different people have different strengths. Individual instructional preferences exist and can be measured reliably. Given suitable environments, resources and approaches, students attain statistically higher achievement and attitude test scores in matched, rather than mismatched treatments. Most teachers can learn to use the concepts of learning styles as a cornerstone of their instruction. Many students can learn to capitalize on their learning style strengths when concentrating on new or difficult academic material (Dunn, 1990).

The University of Newcastle-upon-Tyne carried out two projects to evaluate the models of learning styles inventories and their impact on post-16 pedagogy (Coffield et al., 2004). Their main questions were which models are influential or potentially influential, and what is the empirical evidence to support the claims made for these models. They reviewed Dunn & Dunn's model, and conclude that "Despite a large and evolving research programme, forceful claims made for impact are questionable because of limitations in many of the supporting studies and the lack of independent research on the model" (Coffield et al., 2004, p.35).

2.2.7.3 Learning Styles in practice

Moving to the practical problem of implementing the concepts of learning styles for web applications, Stash, Cristea and Bra (2004) looked at the application of learning styles in the new educational space created by the Web. They wanted to provide authors with a tool that enabled them to use different learning models in their own adoptive educational hypermedia. Table 2.1 shows how, according to Stash et al. (2004), the main existing systems relate to students' learning styles.

System	Learning Style
AEC-ES (Triantafillou et al., 2002)	field-dependent (FD) and field-independent (FI) style
ARTHUR (Gilbert et al., 1999)	visual-interactive, auditory-lecture and text styles
CS388 (Carver, 1999)	Felder-Silverman (1988) learning styles model, global-sequential, visual-verbal, sensing-intuitive, inductive-deductive styles
INSPIRE (Grigoriadou et al., 2001)	Honey and Mumford (1992) categorisation of activists, pragmatists, reflectors and theorists based on Klob
iWeaver (Wolf, 2003)	auditory, visual , kinaesthetic, impulsive, reflective, global, analytical styles of Dunn and Dunn's (1990) learning style model
MANIC (Mia, 2000)	applies preferences for graphic versus textual information
Tangow (Paredes et al., 2006)	sensing-intuitive dimension from the Felder-Silverman (1988) learning style model

Table 2.1: Learning styles incorporated into adaptive systems (Stash, Cristea, Bra, 2004)

Stash et al. (2004) did not recommend any particular instructional strategy for a particular learning style. Their work focused on implementing various instructional strategies and providing authors with tools that allow them to define adaptive strategies and specify which instructional strategies should correspond to which learning style.

One of the systems listed in Table 2.1 is iWeaver, which suits the auditory, visual, kinaesthetic, impulsive, reflective, global and analytical learning styles of the Dunn and Dunn (1990) learning style model. It was a PhD project designed by Wolf (2003) to provide a flexible and manageable environment for the learner by implementing adaptive hypermedia techniques. Its importance here is that, like this research, it applies its strategies on computer science students in higher education, using the Dunn and Dunn (1990) learning styles model.

A statistically based study by Chamillard and Sward (2005) shows that a student's learning style not only affects his or her performance in introductory computer science courses, but that it can also affect performance across all the courses in the CS curriculum. Although they found a wide variety of statistically significant results, it was unreasonable to expect significant results across all course assessments for all dimensions of the various learning style models included in the case study. The wide-range of results and the limitation of the sample, as all the tested students were from one university, means that there should be caution about generalising the results to all CS students. Chamillard and Sward themselves suggest that further studies are required, and that the approach would better if it were course-specific rather than covering the whole curriculum. The effect of implementing learning styles cannot always be clear even with statistically based studies.

A later work on learning styles tried to address the effectiveness of considering learning styles in computer science courses and the cultural differences between students. Zualkernan's (2006) hypothesis was that cultural background may impact on learning styles and those patterns of thinking may differ from one culture to another. However, his investigation concluded that there are strong similarities in learning styles among students from different cultures. This supported Howard's et al. (1996) study, which concluded that there is great similarity in learning styles between students from different cultures, and the comparative learning style profiles mirrored one another in almost all respects. However, More's (1989) comprehensive survey of multiple studies shows that Native Americans differ in their learning styles from people of Caucasian descent; the former tend to be more visual than verbal and more reflective than active in their learning styles. The adaptation of teaching and learning systems of students' learning styles in any learning environment is crucial to accomplish the optimal collaboration between the student and the learning materials. Moreover, such adaptation creates a positive relationship between the learner and the teacher, as the student feels better able to absorb information, and to keep up with the progress of the class or lecture. Previous work has shown that a consideration of learning styles is very important to the learning process, and it has presented some implementations that tried to help CS students to achieve the best results when learning.

2.2.8 Cognitive Load

The cognitive load theory (CLT) was introduced by Swiller in 1988 (Kalyuga, 2006). CLT is defined as a model for instructional design based on the knowledge of how learners acquire, process and retain new information (Sweller, 2008). It proposes that a successful use of the model results in more effectual learning, and greater retention of information in the long term memory, so that it can be recalled when required (Seery, 2008).

In computer science modules, students are exposed to different visualisation tools which are used for teaching programming concepts. The use of multiple visual tools, available on the internet, to help students to understand data structures concepts, increases the cognitive load on students' brains (Tudoreanu, 2003). Section 2.6 will discuss in more detail the relationship between cognitive load and the use of visual learning tools. It will show that the literature suggests that the use of visualisation tools does not necessarily have a positive impact on learning. However, the use visualisation together with audiolisation does tend to lead to a better learning experience.

2.2.9 Summary

The research covered in Section 2.2 shows the importance of interactive learning environments, and the necessity to have the appropriate tools to facilitate students' interaction as well as the knowledge to use such tools professionally and skilfully. Moreover, it is necessary to be aware of student's perceptions, and their willingness to accept these new approaches.

2.3 Learning and Teaching data structures

2.3.1 Why data structures?

This section explores the importance of OOP for CS students. It looks at the feasibility of learning object and data abstraction even before starting computer science studies at university. It investigates this in relation to project management skills, and solving real life problems.

Early in the 1970s, the need for software engineers to produce large software systems increased (Olthoff, 1986). The complexity of these new programs and the facilities offered by procedural languages made it very difficult to keep track of the coding process and error correction. Size and complexity also reduced the reliability and readability of the programs. The OOP concept and data structures emerged early in the 1980s; OOP aimed at going beyond structured programming to gain an understanding of its elementary principles and properties, while data structures advanced the idea of dividing the problem into smaller pieces, and creating reusable building blocks called objects.

2.3.2 First Year Students and OOP

Although the necessity of having an OOP language as a starting point for first year CS students was understood at an early stage, the use of the language was not easily achieved in practice (Decker and Hirshfield, 1995). Both students and staff struggled during the learning process because there was limited knowledge of the OOP domain among teaching staff and few resources were devoted to the problem. In addition, it was feared that adopting OOP as the programming concept for first year students would cause serious harm to the entire CS curriculum and necessitate a restructuring of the introductory programming sequence.

However, the advantages of OOP led many researchers (e.g., Deckter, 1995; Kolling, 1995; Rajaravivarma, 2003) to study the usability of programming languages (PL) and their role in building the basics of OOP in students' minds. Also, interest in teaching OOP rose significantly in the 1990s and many researchers looked at applying OOP concepts to the existing PLs. Kolling (1995) examined the deficiencies of existing languages like C++, Smalltalk, Eiffel and Sather. He found that even C++ has some features that clash with OOP concepts. His work convinced him of the importance of OOP in computer science studies and, consequently, he suggested creating a new programming language that would meet its requirements and he listed some features that it might have.

In the late 1990s there many attempts to create the best tutoring system for CS students and these emphasised the ability to visualise the concepts of data structures and their simple manipulation methods (Shilling, 1992; Warendorf, 1997; Pieson, 1998; Hansen, 1998). This led to a focus on creating interfaces to help CS students in their data structures courses. Warendorf (1997) created the Animated Data Structure Intelligent Tutoring System (ADIS) to help students to learn data structures easily and quickly. It was straightforward software to visualise the main operation on some of the data structure methods, such as the creation, addition and deletion of objects in simple classes.

2.3.3 OOP in Java

The concept of data structures was introduced through the existing PLs that support OOP. The current focus is on how to use Java as a powerful PL that can clearly introduce OOP and its features. Java was created by Sun Microsystems in 1991 and published in 1995[1].

According to Brosgol (1998), at that time, Java was not the best choice for teaching the concepts of OOP. He argued that Ada might be a better choice than Java as a foundation language in CS education because of its learnability. But, soon after 1998, Java proved its capability to implement more complicated OOP through its rich standard application programming interface, which helps students to write applications involving networking, multithreading and many other techniques without having to resort to non-standard, third-party libraries (Schaub, 2000). However, it still lacked a proper graphical user interface (GUI), which made it hard to learn the coding techniques and programming concepts easily, although better programming interfaces do not necessarily provide a better programming language. Schaub (2000) advocated the use of the old Turtle API from Seymour Papert's Logo. He believed the Turtle paradigm would be a good way to introduce basic programming concepts since many OOP concepts can be presented using simple graphics, thus improving the ability of students to understand those concepts, and use them effectively.

Universities then had the challenge of using Java to introduce OOP concepts as a CS foundation course and, at the same time, to stay within the time limit imposed by the term structure. Rajaravivarma and Pevac (2003) presented a new approach that introduced the concept of Objects at the beginning of the course, and followed that with an emphasis on problem solving techniques. They suggested that the use of real world examples would enrich students' understanding of OOP concepts and enable them to implement them easily in later, advanced courses in computer programming. However, there is a continuing debate about the learning

[1] (www.java.com)

gain that can be achieved by using OOP at early stages of CS courses compared to using it at later stages (Ehlert, 2009).

OOP teaching and the understanding of data structures had a high priority for researchers and module leaders in higher education. They pursued the idea of using any available resource for teaching these subjects in a way that was both effective and that dealt with the higher demands of studying computer science. This resulted in using the World Wide Web (WWW) and multimedia tools. In an attempt to support staff in teaching CS modules, Daly and Horgan (2004) developed RoboProf, an automated learning environment that generates and assesses programming exercises, and provides ongoing assistance and feedback to students without extra demands on lecturer time. This and other such applications aimed to teach students the basics of programming and to facilitate their own control over what they were learning and their time commitment, thus giving students another resource of learning that they might use in their own time to develop a deeper understanding.

Vickery (2005) looked at the feasibility of teaching OOP at GCE 'A' Level using Java, especially the use of an Integrated Development Environment (IDE) called BlueJ. His interest came from the idea of the importance of knowing how to program using object oriented principles at school level to ensure a smooth transition to higher education courses. His work looked at the use of procedural PL but this showed limitations, even with the use of a visual PL like Visual Basic or Delphi. Throughout the experimental work, carried out by an Information and Communication Technology (ICT) teacher, the students appreciated the instant visual feedback given by BlueJ, which bridged the gap between the theory behind OOP and its visual interpretation. However, the study found that the scheme was only good for learning Unified Modelling Language (UML), and as an introduction to concepts, but not useful enough as a programming tool (Vickery, 2005).

2.3.4 Evaluation of data structures teaching and learning tools

The use of visual learning tools to help students to build a mental image of how OOP works is one of many attempts to use technology to improve learning. Many researchers have developed visualisations and tutoring tools to enhance students' learning experience and open students' minds to engage with and understand the concepts rather than relying on memory (Brown, 1991; Shilling and Stasco, 1992; Culwin, Adeboye and Campbell, 2006). But, auditory interfaces can enhance visualisation tools and thereby support students' engagement (Higgins, 2000). This section will look at previous studies on supporting student engagement with learning resources and discuss their findings.

There have been many trials that aimed at enhancing the teaching of CS students by providing lecturers and students with a variety of tutoring systems (Moons, 2009). Those systems can be classified, according to their target objectives, into three categories, namely, expert IDE environments, micro-world environments and advanced visualisation environments.

2.3.4.1 Expert IDE Environments

An Integrated Development Environment (IDE), also known as an integrated design environment, can be defined as an enclosure of the principle development tools under a single, consistent user interface that helps the development process of a computer programme by automating some of the time consuming elements such as the graphical user interface (Depradine, 2004). Examples are jGrasp, BlueJ, and Greenfoot (Cross, 2004; Kölling, 2006; Kölling, 2008).

BlueJ's main visual features are the display of the class structure in the main window and the visual distinction between classes and objects; the latter is an important issue and one that is difficult to explain. Those features have influenced, in this research, the design of objects representation in the "Learn About Objects" visualisation of the DSL tool. However, BlueJ probably has a limitation as a beginner's tool. It focuses on the object-oriented paradigm for modelling and design by relying on the class diagram as its basis structure, and it does not provide an effective presentation of data structure at execution time (Moor and Deek, 2006). Greenfoot, created by the same people who created BlueJ at the University of Kent, is an educational IDE that provides a set of learning tools that aid in understanding basic object-oriented concepts, and it is considered as a motivational environment because of its ability to provide instant graphical feedback (Kölling, 2008).

Another example of an IDE environment is JGrasp, which was created specifically to provide automatic generation of visualizations in order to improve software understanding. The visualisations it produces include Control Structure Diagrams, UML Class Diagrams, and dynamic Object Views. For example, visualizations for more complex structures, such as linked lists and trees, are generated as needed from the user's actual program during routine development (Cross, 2004).

Each of these IDEs, BlueJ, Greenfoot and jGrasp, provide learning environments that aid students in getting used to interacting with object instances and exploring their behaviour in an uncomplicated approach. Moreover, these learning tools are recommended by Sun Microsystems for users without programming experience and listed as tools designed to demonstrate programming visually instead of just looking at lines of confusing code (Nourie, 2008).

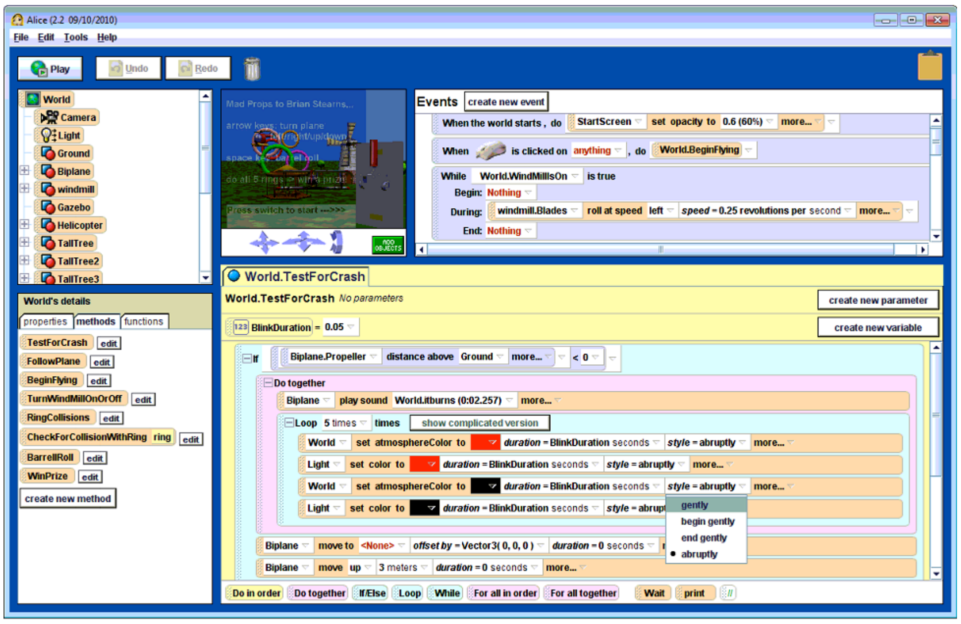
2.3.4.2 Micro-worlds Environments

Micro-worlds can be defined as self-contained computer-based virtual programming environments in which students have the chance to explore, interact and establish learning activities to learn basic concepts (Wilson, 1995). It is a learner-centred world explored by directly manipulating its objects with a small number of basic commands. Micro-worlds can be joined with images and diagrams to aid in describing programming problems and to make use of a storytelling approach as an educational paradigm (Gross, 2005).

Alice (www.alice.org) is an extensive micro-world with a non-Java syntax. It is a 3D programming environment with a functionality to allow users to drag and drop objects in its virtual world and then add methods or actions that the object can perform (Briana Lowe et al., 2009). Alice also allows the direct manipulation of added objects' values through inspection of its values or scripting in the Python programming language. However, although Alice can easily be programmed into object instances, it lacks any particular task structure (Culwin, 2006).

Barbra et al. (2004) examined the effectiveness of using Alice in a first year computer science course to improve the performance of low scoring or "at risk" computer science students. They found that the Alice course improved novice students' performance, and improved their attitudes towards computer science. The study also found that high risk students who participated in the Alice course showed high retention rates while high risk students who did not participate showed signs of low retention rates (Barbara, 2004).

Although these advantages can be claimed for Alice, it has some pedagogical pitfalls. The object model in Alice can easily lead to a false impression that programming is very easy. The lack of warnings about syntax errors can raise students' confidence unjustifiably and cause problems when they come to code in other programming languages, like Java (Gross, 2007). Figure 2.3 shows a worked example of Alice micro world. This example teaches students how to programme mouse events, loop, and setting objects properties.



```
When the world starts , do   StartScreen set opacity to 0.6 (60%) more...
When [mouse] is clicked on anything , do   World.BeginFlying
While World.WindMillsOn is true
  Begin: Nothing
  During: windmill.Blades roll at speed left speed = 0.25 revolutions per second more...
  End: Nothing
```

```
World.TestForCrash No parameters
123 BlinkDuration = 0.05
If Biplane.Propeller distance above Ground more... < 0
  Do together
    Biplane play sound World.itburns (0:02.257) more...
    Loop 5 times times
      World set atmosphereColor to  duration = BlinkDuration seconds style = abruptly more...
      Light set color to  duration = BlinkDuration seconds style = abruptly more...
      World set atmosphereColor to  duration = BlinkDuration seconds style = abruptly more...
      Light set color to  duration = BlinkDuration seconds style = abrupt
  Biplane move to <None> offset by = Vector3( 0, 0, 0 ) duration = 0 seconds
  Biplane move up 3 meters duration = 0 seconds more...
```

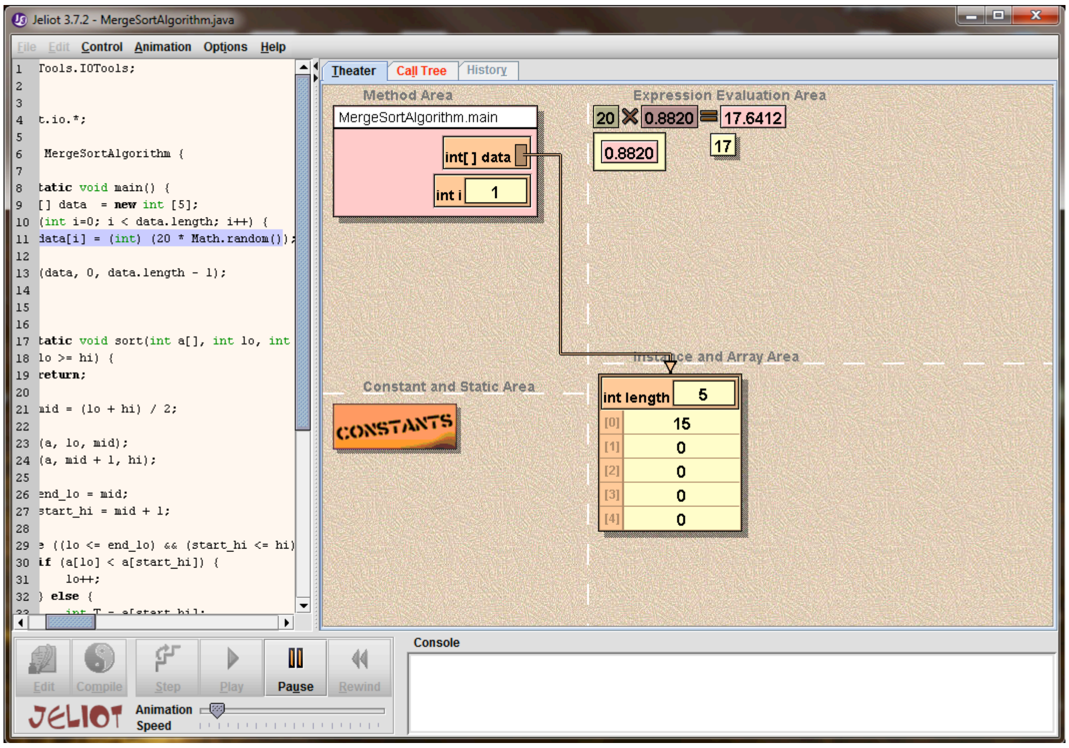
gently
begin gently
end gently
● abruptly

Figure 2.3: Snapshot of Alice 2.2 Microworld

2.3.4.3 Advanced Visualisation Environments

Advanced visualisation environments referred to in this research have some similarity to visualisation targeted at novice programmers. Such environments can use simultaneous representation of visualisation, and present data as it is processed by a virtual machine, such as JEliot 3 (Bednarik, 2006), or show the program runtime state of object instances, such as Seppälä's (2004) program state diagram. These advanced visualisation environments provide integration with, or can be 'add ons' to, IDEs like BlueJ, Eclipse or the integration of EduVisor in Sun's Microsystems development environment, Netbeans (Moons, 2009).

JEliot 3 is a research-oriented visualisation tool that was designed to aid novice students to learn procedural programming and OOP with either fully or semi-automatic visualization of the data and control flows (Moreno, 2004). It claims to have a simple interface with complete visualisations that are self-explanatory for students. It can be extended to be integrated with a third-party environment (BlueJ), so that students who use BlueJ to study Java can find it useful. Moreover, it supports three architecture models of Java programs, namely, having the ability to write code, to visualise its execution and to trace changes. However, some studies have found that it has some shortcomings. Maravic et al. (2010) conducted a study where 45 students were tested on their comprehension of short program segments. His results suggest that students who learned with JEliot 3 made detailed, concrete mental representations of the program text and supported it with better test examples than students from the control group. However, some students regarded it as a waste of time. They complained about its simplicity, felt that it was helpful only for beginners and thought that the visualisation caused distraction. JEliot 3 also has some usability issues as its canvas is based directly on the Java AWT classes and this causes some restrictions, such as the complete absence of select, zoom and pan tools when manipulating onscreen visualisations (Moons, 2009). A working snapshot of JEliot 3, presenting the visualisation of the merge sort algorithm, is shown in Figure 2.4.



```
Tools.IOTools;


t.io.*;

 MergeSortAlgorithm {

tatic void main() {
[] data  = new int [5];
(int i=0; i < data.length; i++) {
data[i] = (int) (20 * Math.random());

(data, 0, data.length - 1);



tatic void sort(int a[], int lo, int
lo >= hi) {
return;

mid = (lo + hi) / 2;

(a, lo, mid);
(a, mid + 1, hi);

end_lo = mid;
start_hi = mid + 1;

e ((lo <= end_lo) && (start_hi <= hi)
if (a[lo] < a[start_hi]) {
    lo++;
} else {
    int T = a[start_hi];
```

Figure 2.4: A snapshot of Bednarik's JEliot 3 visualisation tool

As explained in this subsection, there have been many trials and much research to try to create the optimal visualisations tool. Many useful tools have been created and used over the past decade, and they have attracted significant interest from both lecturers and students. However, the created visualisations have been frequently criticized by other researchers in the field. This research, on the other hand, did not try to add extra features or to create more complicated visualisation and fancy animation. On the contrary, it aims to create a focused learning tool to help students by providing usable visualisation with the least distraction and the least possible cognitive load on their working memory.

2.3.5 Summary

Section 2.3 has covered an important part of this research. It has discussed how OOP development tools have improved and adopted the new learning techniques over the years. OOP and data structure concepts are now the basis of any student programming module. Java is currently the preferred programming language and, in order to create a learning tool, it should be considered as the presented coding syntax. It is also important to know about the latest OOP development tools, the reasons for their creation, their visual elements, and their benefits and disadvantages. Section 2.4.3 will discuss, in more detail, a variety of such tools that have been developed to assist students in learning programming concepts.

2.4. Algorithm Animation and Visualisation

2.4.1 Definition

Visualization can be defined as the mapping of data to representations that can be visual, auditory or a combination of other means of communication. It is a method of computing that transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations (Owen, 1999).

Algorithm animation is a form of program visualization that is concerned with dynamic and interactive graphical displays of a program's fundamental operations (Brown, 1991). Animating algorithms started as an exploration of using high-powered processors to provide complex visual representations of the behaviour of data structures as they were acted upon by algorithms (Gurka, 1996).

2.4.2 Why Visualisation?

Visualisation is a method of using technology to improve learning by creating a mental image of how things work (Messner, 2003). It is also a learning style that many students prefer as a means to increase the comprehension of concepts, bearing out the proverb that “a picture is worth a thousand words.” It also increases the interaction between the learner and the subject being studied, so it is an active learning approach.

In computer science, visualisations help researchers to recognise patterns in big data sets that are difficult to see by using only textual and mathematical representations of data (Guzdial, 1994). Computer science educators find algorithm and data structure visualizations useful in their classrooms. However, research shows that some are effective while many are not (Cooper, 2007).

Cooper’s work involved collecting all the visualising algorithm implementations in one wiki website, Algoviz Wiki. This helps students to find an existing visualisation to help them in their computing studies, and it helps to create ideas for new visualisation tools as it shows what researchers have not presented properly or visualisations that have not yet been created. Figure 2.5 shows some statistics on the availability, in 2007, of visualisations for different purposes. It shows that most of the visualisations were for sorting algorithms, search structures and linear structures. This result reflects the importance of those structures and the need for students to visualise them to help them to understand the subject. The results also show that few visualisations have been created for search algorithms and memory management. Search may not need to be visualised as it is easy to understand and it is already part of the sorting process.

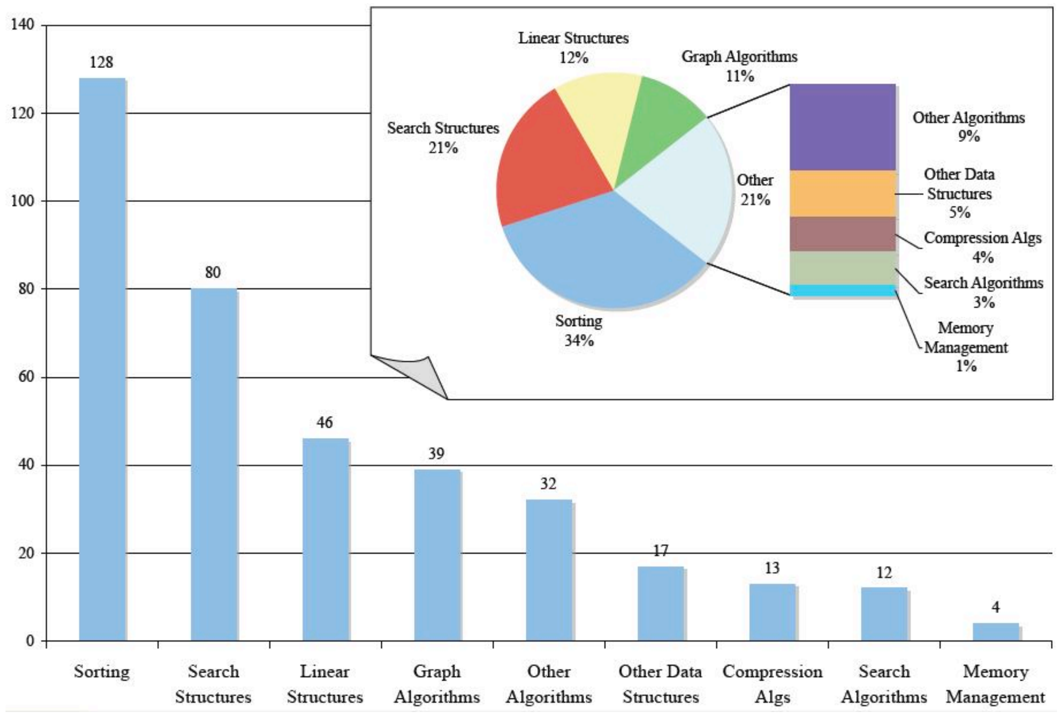


Figure 2.5: Cooper's distribution of visualizations by topic (Cooper, 2007)

The use of graphics in higher education has a long history but only in the 1980s did it become a focus of research. One example is Zeus, Brown's (1984) visualisation software. He stated that "it is possible to expose the fundamental characteristics of a broad variety of programs through the use of dynamic (real-time) graphic displays and that such algorithm animation has the potential to be quite useful in several contexts" (Brown, 1984, p.1). Zeus is mainly concerned with animating algorithms to ease their functional understanding so it helps the learner to have imagination while coding.

Naps (2002) identified a set of 'commandments of algorithm animation'. These suggestions raise some important issues about providing resources that help learners to interpret the graphical representation by adapting to the knowledge level of the user and providing multiple views. Naps also suggests including performance information and execution history and supporting flexible execution control, learner-built visualizations, custom input data sets, dynamic questions and dynamic feedback. Moreover, he proposes complementing visualizations with explanations and he encourages educators to adapt and apply visualisation carefully, since no single tool is the best for all learners. Like any other design activity, the design of animation systems should be carefully planned. This point links back to Section 2.3.4 and the importance of adopting tools that correspond to students' different learning styles, as proposed by Biggs's (2003) notion of constructive alignment.

Byrne, Catrambone and Stasko (1996) carried out two experiments. The first, using a depth-first search algorithm, found that, when learners both viewed an animation and made predictions, their performance on novel problems improved compared to a control group. However, the effects of both animation and instruction were not distinguishable. In the second experiment, they used a Binomial Heap algorithm and no effect was found for conceptual measures of learning but a slight effect, similar to the one seen in the first experiment, was found for procedural problems.

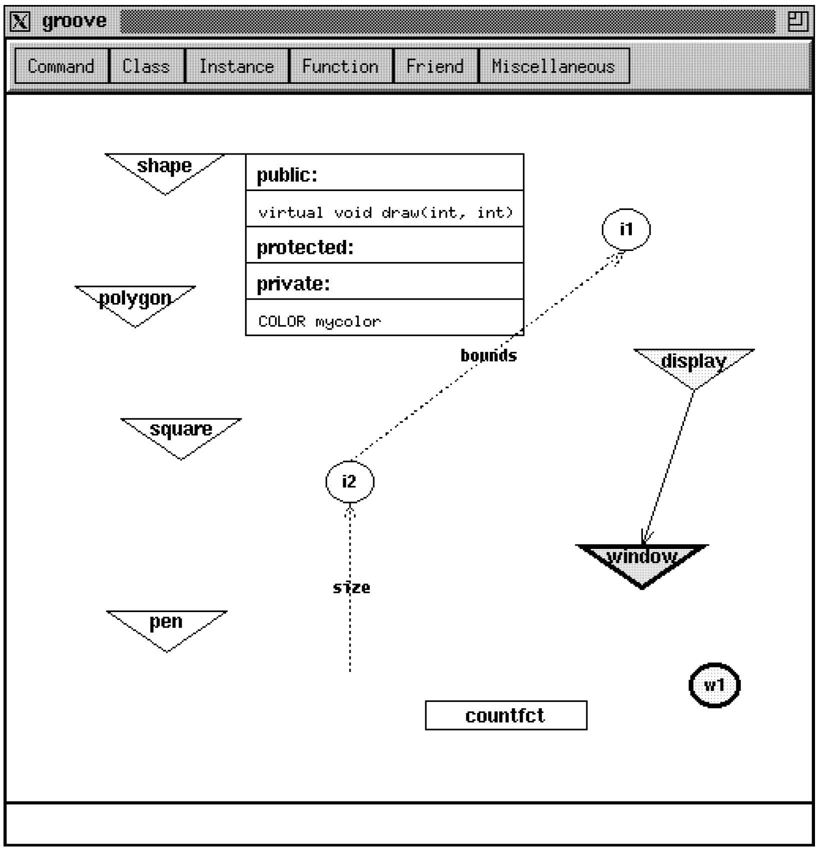
2.4.3 Overview of existing visualisations

The following sub sections examine some popular CS visualisation tools. The tools selected for discussion in this section were chosen for their focus on visualising data structures and OOP.

2.4.3.1 GROOVE: 1992

GROOVE (Graphical Object-Oriented Visualization Environment) is a visual design tool that allows programmers to visually specify both the static structure of a program and its run-time dynamics and protocols. It is designed to help programmers to visually specify both static structure and dynamic protocols of an object-oriented program. GROOVE's visual paradigm employs shape, colour and animation to portray objects and relationships. It helps developers understand message-to-method binding by utilizing the class presentations from which an object inherits fields (Shilling and Stasco, 1992).

Figure 2.6 shows GROOVE’s designer interaction with the visual presentation and how it automatically generates and updates an accompanying code template, shown in the neighbouring window, which corresponds to the design. This feature can be a valuable educational aid for students learning object-oriented design. Students only focus on the important concepts and they are not be held up by numerous syntactic errors that traditionally accompany learning a language (Shilling and Stasco, 1992).



public:
virtual void draw(int, int)
protected:
private:
COLOR mycolor

Figure 2.6: A GROOVE program specification (Shilling and Stasco, 1992)

2.4.3.2 JAWAA: 1998

JAWAA (Java and Web based Algorithm Animation) is an example of a simple command language for creating animations of data structures and it provides an interface that displays them with any Web browser. It is helpful as an alternative view in understanding newly presented data structures and as an aid in debugging programmes that use the data structures. JAWAA commands can be added as output to any program to quickly generate an animation (Pieson, 1998). JAWAA applies the active learning concept through the interactive lecture format and it can benefit group teaching.

Figure 2.7 shows an example of JAWAA selection sorting for an array. The orange key index shows the selection of the current smallest value and the pink key shows the value to be compared. After comparison, the smallest value will be orange, and the process of comparisons starts again.

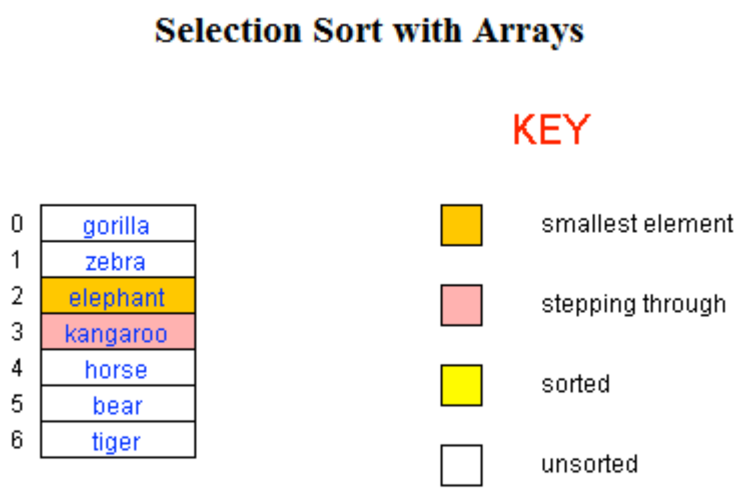


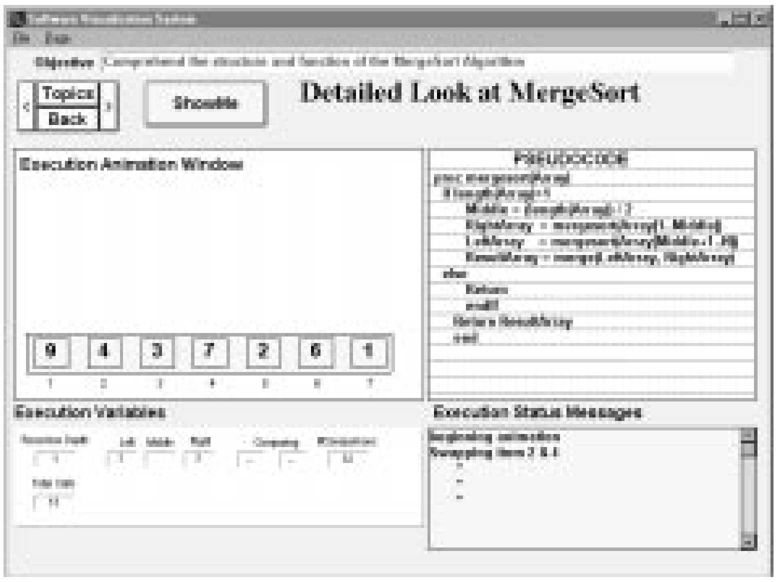
Figure 2.7: JAWAA's selection array sorting example (Pierson, 1998)

2.4.3.3 HalVis: 1998

The HalVis (Hypermedia Visualisation) approach to algorithm visualization is based on how humans reason using static diagrams to infer the dynamic behaviour of mechanical devices and on how hypermedia can improve student comprehension of the design and evaluation of algorithm animations. HalVis allows students to learn incrementally by starting from a real world analogy and transition to the algorithm itself. Moreover, the hypermedia structure allows students to access fundamental building blocks of algorithmic knowledge at any time (Hansen, Schrimpsher and Narayanan, 1998).

Hansen et al.’s (1998) key insight is that, for algorithm animations to be effective, they have to be chunked and embedded within a context and knowledge providing a hypermedia information presentation system. In this process, the problem is presented in stages that start by visualising the big picture and then go down to more detailed visualisations. Experimental tests of the system show that students tend to execute the detailed animation less frequently than the populated-level animations, which are algorithms on large data sets. However, previous research results by Stasko et al. (1993) show that, although the visualisations are received enthusiastically by students, no student showed any improvement in the learning process.

Figure 2.8 is a snapshot from HalVis system. It shows seven data elements to be sorted using the MergeSort algorithm. It moves the seven data items as needed until the algorithm is finished and all elements are sorted.



Topics
Back
ShowMe
Detailed Look at MergeSort
Execution Animation Window
9 4 3 7 2 6 1
PSEUDOCODE
Execution Variables
Execution Status Messages

```
proc mergesort(Array)
  If length(Array) > 1
     Middle = (length(Array)) / 2
     RightArray = mergesort(Array[1..Middle])
     LeftArray  = mergesort(Array[Middle+1..N])
     ResultArray = merge(LeftArray, RightArray)
  else
     Return
     endif
   Return ResultArray
   end
```

```
Beginning animation
Swapping items 2 & 4
```

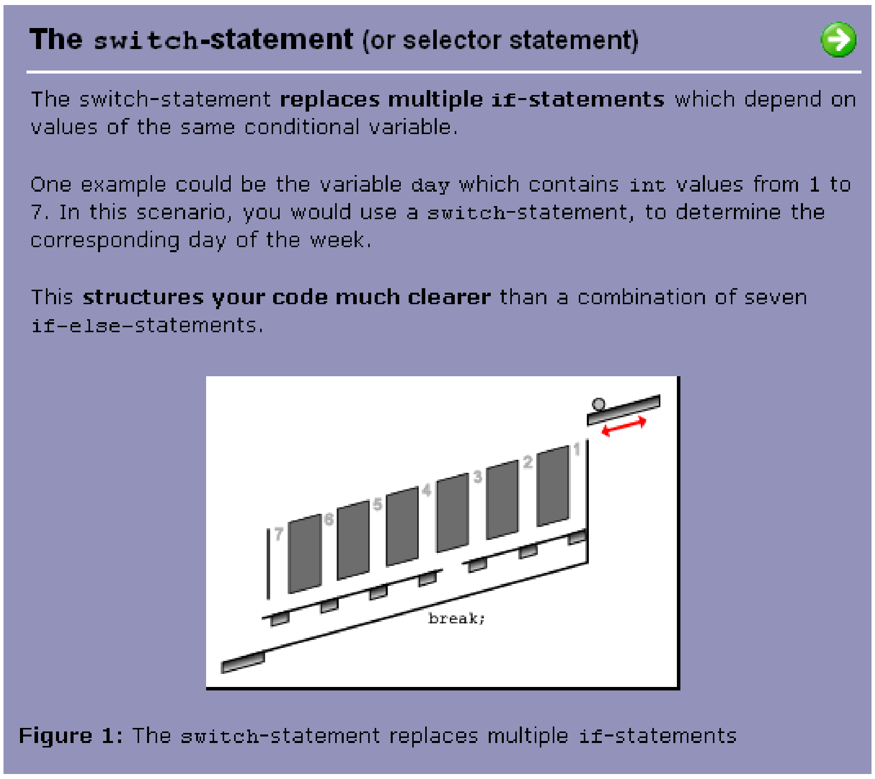
Figure 2.8: HalVis detailed view screen (Hansen, Schrimpsher, Narayanan, 1998)

2.4.3.4 iWeaver: 2002

iWeaver is an interactive web-based adaptive learning environment. It is built on Dunn and Dunn’s (1990) learning style model, which assumes that each learner has an individual learning style. It encompasses two dimensions: media experiences and learning tools. The iWeaver system identifies a student’s learning style by asking each user to answer 120 multiple choice questions and, based on the Dunn and Dunn (1990) model, it adopts a suitable learning style for that user. The results are then saved in the user’s own profile so the student does not have to repeat the test.

The system also presents a number of learning tools, aiming to accommodate different psychological styles. The ‘global learners’ can access mind-map diagrams to understand the sense of the big picture. ‘Reflective learners’ are offered the chance to take notes and answer reflective questions. ‘Impulsive learners’ can immediately try out example code with the help of an online compiler. ‘Visual text’ learners experience content in a rich text format, whereas the content for ‘visual picture learners’ is supplemented by additional illustrations, diagrams and animations. ‘Tactile kinaesthetic’ learners are accommodated by ‘interactivelets’ and ‘auditory learners’ encounter a PowerPoint style presentation of the learning content (Wolf, 2002).

Figure 2.9 shows an example of teaching the ’select statement’ procedure, using animated object movements over bars representing available choices, and the object will explain the one selected. Along with visualisation, a pre-recorded voice explains the whole process.



The `switch`-statement (or selector statement)

The switch-statement **replaces multiple `if`-statements** which depend on values of the same conditional variable.

One example could be the variable `day` which contains `int` values from 1 to 7. In this scenario, you would use a `switch`-statement, to determine the corresponding day of the week.

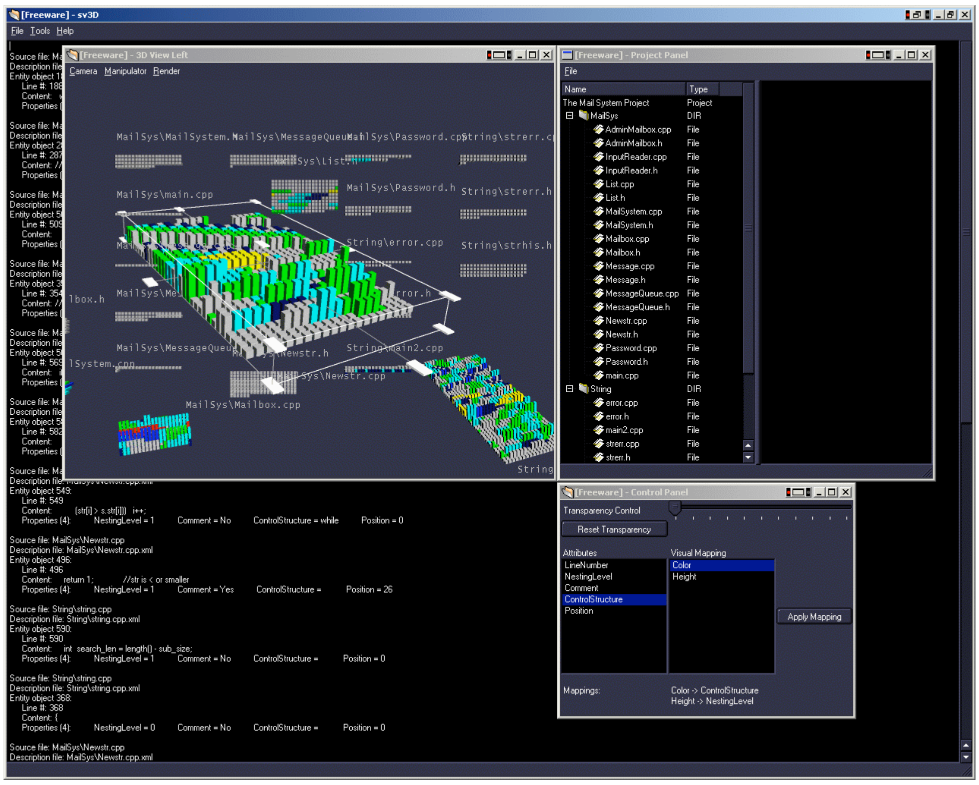
This **structures your code much clearer** than a combination of seven `if-else`-statements.

Figure 1: The `switch`-statement replaces multiple `if`-statements

Figure 2.9: iWeaver media experience (Wolf, 2002)

2.4.3.5 SV3D: 2003

The SV3D (Source Viewer) framework is a 3D metaphor for software visualisation based on SeeSoft's pixel representation and 3D file maps, relying on a visualisation system that can display thousands of lines on a single screen to allow the detection of patterns. It is used to represent source code and related attributes. Using poly cylinders with four edges and uniform fill, it will allow representations of hierarchical data and diagrammatic visualisation, such as UML class diagrams. However, there is a problem in scalability of this 3D space. SV3D is an example of analytical tools that aid the work of researchers in data and file mapping tasks in software engineering applications (Marcus, Feng and Maletic, 2003). Figure 2.10 shows a working example of sv3D framework.



```
Entity object 549:
    Line #: 549
    Content:        (str[i] > s.str[i]))   i++;
    Properties (4):       NestingLevel = 1       Comment = No       ControlStructure = while       Position = 0

Source file: MailSys\Newstr.cpp
Description file: MailSys\Newstr.cpp.xml
Entity object 496:
    Line #: 496
    Content:     return 1;           //str is < or smaller
    Properties (4):       NestingLevel = 1       Comment = Yes       ControlStructure =        Position = 26

Source file: String\string.cpp
Description file: String\string.cpp.xml
Entity object 590:
    Line #: 590
    Content:     int  search_len = length() - sub_size;
    Properties (4):       NestingLevel = 1       Comment = No       ControlStructure =        Position = 0

Source file: String\string.cpp
Description file: String\string.cpp.xml
Entity object 368:
    Line #: 368
    Content: {
    Properties (4):       NestingLevel = 0       Comment = No       ControlStructure =        Position = 0

Source file: MailSys\Newstr.cpp
Description file: MailSys\Newstr.cpp.xml
```

Figure 2.10: sv3D working example (http://graphics.idav.ucdavis.edu/~lfeng/research/sv3d/index.html)

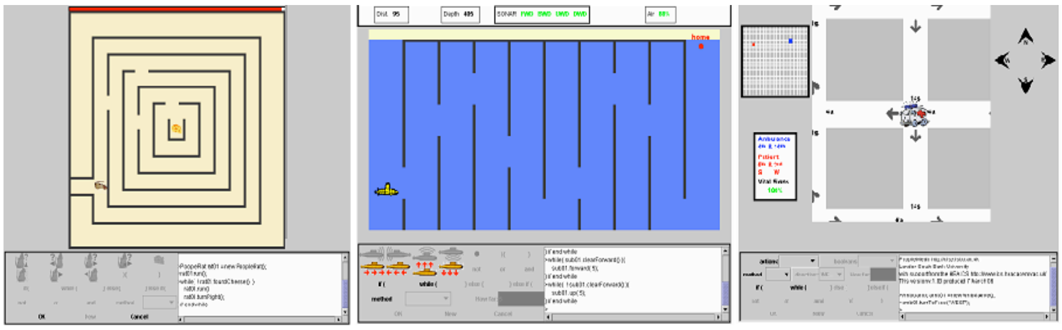
2.4.3.6 SHALEX: 2005

The SHALEX (Structured Hypermedia Algorithm Explanation) system is another good example of algorithm animation software. The system provides a hypermedia environment that can reflect the structure of an algorithm through directed graphs of each abstraction; each one is designed to focus on a single operation, and to provide an abstract data type (ADT) giving a high-level view of generic data structures. The first version of this system was implemented using Macromedia Flash 2003 but an HTML web based format is being considered for the next version, as this will be more accessible to the system’s users. SHALEX supports active learning through its interactions with users: posing various questions about the algorithm to the user, along with a “do-it-yourself” mode which provides the user with a means of testing his/her understanding of the given algorithm by attempting to explain it (Müldner, 2005).

2.4.3.7 POOPLES: 2006

Culwin et al.'s (2006) POOPLES (Pre-Object Oriented Programming Learning Environments) is a prototype visualisation system implemented in three-part software. The first part is "poopRat", the simplest prototype, which depicts a rat having to run through a maze and reach the cheese before it dies from hunger. The second is "poopSub", which involves driving a submarine though a series of baffles before it runs out of air. The final one, which is the most complex because of the way it is controlled, is "poopMedic". This involves driving an ambulance through a regular grid of streets and avenues to reach a patient before he dies. Several universities have used POOPLES in their classes, especially when presenting their programming materials for the first time. The system was evaluated with school students who aspired to study computing or information technology at university and the results indicated that participants were highly engaged in the environment as they were operating independently of the tutors and very task focussed. This was supplemented by the production of support material directing the students' attention to the control structures and the use of methods. Although the system was evaluated using very small studies, the results show it is beneficial for students who have yet to start a university programming course, and that it is an additional resource to convince them that programming can be fun. However, the results also show that students who already disliked programming were not re-motivated by the experience (Culwin, Adeboye and Campbell, 2006). Figure 2.11 shows snapshots of three implementations of

POOPLE, poopRat, poopSub and poopMedic. The controls of the visualisation correspond to a Java code posted in the interaction pan.



```
PoopeRat rat01 = new PoopleRat();
rat01.run();
while( !rat01.foundCheese() )
  rat01.run()
  rat01.turnRight();
// endwhile
```

```
}// end while
while( sub01.clearForward() ){
  sub01.forward( 5);
}// end while
while( !sub01.clearForward() ){
  sub01.up( 5);
}// end while
```

Figure 2.11: Culwin's three POOPLEs "poopRat, poopSub and poopMedic" (Culwin, Adeboye, Campbell, 2006)

2.4.4 Summary

Visualisation is a primary focus of this research. Studying existing visualisations can help in creating effective new visualisations. Because of the importance of this field of study, there have been many visualisations designed to aid researchers and students across many fields of study. Cooper estimated that, in 2007, there were over 350 of them, many of which are individual applets or programs, but significant proportions are parts of integrated visualisation collections (Cooper, 2007).

This research primarily investigates the integration of aural instructions in an interactive visualisation environment to create a focused learning environment that leads to a reduction of cognitive load in students' learning processes. The next section will describe auralisation and how it is used in learning.

2.5. Audio-based Interfaces / Algorithm Auralisation

2.5.1 Definition

Aural support or auralisation involves the use of any sound to support an application, either by spoken audio (narrative) or computer-generated sound. According to Brown (1992), algorithm auralisation is considered only when the generated sound is data driven. Early usage of sound in algorithm animation/auralisation depended on generating a short theme while an algorithm was running and another one when it stopped. However, in Brown's study there will be no difference between running two different algorithms regardless of their features in terms of the sound produced

2.5.2 Background

Visualising algorithms is used to enhance learners' experience and facilitate the understanding of an algorithm and its concepts. However, many of the existing visualisations have problems. These include the loss of focus when the abstract representation concentrates on low-level steps rather than on high-level properties like invariants (Müldner and Shakshuki, 2004). This leads to a higher cognitive load on the students' learning process (Tudoreanu, 2003).

The use of audio in computer science and in algorithm animation started as descriptions of what the visualisation was currently showing (Brown, 1992). However, the complexity of some algorithms needed new applications if sound usage was to be effective. One of Brown's early trials was using sound to get the attention of the person running an algorithm. For example, a car crash sound was used to represent the idea of a collision. He also encouraged the use of Buxton, Gaver and Bly's (1991) recommendations of using auditory displays and real life sounds to complement the use of visualisations.

2.5.3 Using Music

It is not clear when using audio to aid programming courses started. However, there have been many trials and wide research studies, starting from the use of descriptive sound to enhance the users interface and convey rich meaning, and continuing with the use of sounds to benefit visually challenged people (Alty, Rigas, Vickers, 1997). However, this approach faced many challenges with screen readers. Vickers et al. implemented a number of sorting and path following algorithms using musical representation alone without supplementing the visual output with sound. They also used algorithm auralisation during the debugging process, and in finding faults in the program execution.

Franklin (2001) used computer-generated music in an introductory programming course as a theme of its projects and also as a means of demonstrating methods for algorithms. The implementation of this work was in a form of two musical projects assigned to students in the introductory course.

The first of Franklin's (2001) projects was the "Ear Trainer", a challenge game or quiz in two parts. First, two different tones were played and the student had to say which was higher. Then, after a tone was played, the student had to identify its pitch. The project focused on if-else statements, switch statements, parameter passing, loops and using user input to cause a program to vary its execution.

Franklin's (2001) second project was the more complex "Song Sorter" that focuses on manipulating arrays by using sort and search methods and enhancing it further by using modulation concepts and pointers. The task in this project was sorting the songs by their number of rhythmic changes and then repeatedly searching for a particular song with specific changes.

At Loughborough University, another relevant study (Vickers and Alty, 2005) tried to show the usefulness of using music auralisation in debugging errors. They found that music could convey information about program events and play a complementary role in the programming process. Although they did not address the needs of visually challenged people, they hoped that their system would be adopted and extended to do so.

Computer generated music is designed as an artistic way to speed up the learning process; for instance, in teaching the theory of computation, regular and context-free grammars can be used to characterise different processes underlying musical improvisation. Moreover, it might be especially interesting for non-CS students who take an introductory programming course because they are interested in an application of computing.

2.5.4 Spoken language processing

The idea of using audio to enhance user interfaces has developed rapidly. As well as progress in using music, work has been done on developing a natural speaking output to enhance the user's interface. In some cases, this enhancement replaced visual information by aural information to help visually challenged people. The opportunity to have a human-type conversation with a computer has been promoted by science fiction films such as HAL in 2001: A Space Odyssey and Star Ship Enterprise (Michael, 2002). As Lai (2001, p.66) says, "Almost two generations of science fiction and movie fans have been raised with the concept of a computer that not only understands every nuance and word in the spoken language but also reads lips, generates flawless speech, and thinks for itself."

Due to improvements in computer capabilities and its popularity with people from different backgrounds, speech and sound technologies have become common methods of communicating with technological appliances; they are integrated into our daily life and people to talk to their computers, watches... etc.. (Stifelman, 1995). Stifelman presented Conversational VoiceNotes, a tool to support the generation of speech and non-speech aural feedback. It uses two output types, speech and non-speech, two input types, speech and buttons, and two levels of detail, brief and verbose. The software feedback varies depending on the user's preferred output modality, the user's preferred detail level, the input modality employed and the time elapsed since the last user command.

Steve Whittaker (1994) presented a novel application that integrates handwriting and recorded audio in a semi-portable device. He proposed rethinking the 'how' and 'why' of audio applications and the use of speech as data, arguing that conversational speech is critical in the workplace, but we do not currently capture it through our computers. He also discussed the importance of recording, accessing and manipulating recorded speech data, even without the power of full speech recognition. In his software Filochat, he provided a user-centred indexing for random access to conversations by co-indexing the digital notes and speech recording.

Recent studies in the field of text-to-speech (TTS) technologies have focused on designing help systems because what we actually need from this technology is to help students to overcome a problem when visually displayed help is not possible, or to enhance the system's accessibility for disabled users. Kehoe and Pitt (2006) suggested a number of guidelines to assist in the creation and testing of help material that may be presented to users via speech synthesis engines. They pointed out a number of choices to be made with respect to the selection and use of speech technologies: Recorded Speech versus TTS, Voice and Persona, Testing and Localization. Their system retrieves help topics and presents them to the user as synthesized speech. The synchronized non-voice audio is also played in parallel with the synthesised speech so that the user receives additional cues. This sonification is used to provide information about the help topic structure, for instance, topic titles. By testing pre-existing help topics from Microsoft Windows applications using speech technologies, they encountered some problems involving the content and the format of those systems. A set of guidelines, based on their findings, can be used when authoring help materials using speech technology. Those guidelines were based on having interactive speech dialog in help systems, as well as customising speech output to the users.

2.5.5 Spoken language software and its uses

Interesting work carried at the State University of New York College by Higgins (2000) looked at the development of speech-enabled programming applications in the undergraduate computer science curriculum. He wanted to show that speech enhanced interfaces can be integrated into the CS curriculum at the second year level and later reinforced through special topics courses, speech enabled projects in Software Engineering, or senior thesis work.

Higgins' work was implemented in two CS curriculum projects on data structures; a stack based calculator and a map or network that implements a graph requiring determination of the shortest paths. In both cases, the solutions for these problems were migrated to a speech-enabled setting and re-evaluated providing very simple graphic user interfaces. IBM's ViaVoice software was used as an input technology to generate the text. Higgins expected that such software would become widely available, make it possible for persons with motor or sight disabilities to use sophisticated programs, and prove useful for anyone who is busy using their hands or eyes to do something else at the same time.

A recent example is the “Almost Realistic” TTS technology for customer service answering systems in mobile phone companies. These answering systems became so real that they have names like Vodafone’s Vicky, and Orange’s Jane. Vodafone went a step further than its competitors by creating Vicky with a virtual persona symbolising the company’s UK’s brand essence and personality. After its introduction, customer satisfaction rates rose dramatically, according to Melanie Rowland, Head of Self Service and Automation-IVR at Vodafone UK. Vodafone planned to automate 60% of all inbound traffic by 2008 (Durant, 2007).

2.5.6 Synthesised Speech

Recent research in speech synthesis has focused on the personality of the generated speech and its effect on users’ perception. In an attempt to create adaptive conversational user interfaces, Microsoft presented a reasoning architecture for an agent that can recognize a user's personality and emotional state and respond appropriately in a non-deterministic manner (Ball and Breese, 1998). They described architecture for constructing a character-based user interface using speech recognition and speech generation. This architecture uses models of emotions and personality encoded as Bayesian networks. The idea was to diagnose the emotions and personality of the user, and generate appropriate behaviour by an automated agent in response to the user's interaction.

Microsoft's suggested architecture is an agent that maintains two copies of the emotion/personality model. One is used to diagnose the user, the other to generate behaviour for the agent. To achieve this adaptation of the conversation, they have followed five procedures: Observe, Update, Agent response, Propagate and Generate Behaviours.

2.5.7 TTS with Visualisation

An interesting study by Qiu and Benbasat (2005) introduced the use of TTS voice with 3D avatar in an online shopping system. They highlighted the fact that, in their help system, the use of interactive visual support with high quality audio enhanced the feeling of telepresence between an individual and customer service. Their research concluded that the use of text-to-speech voice technology has significantly improved customers' awareness and contributed to "cognitive enjoyment and focused attention" (Qiu and Benbasat, 2005, p: 351). These findings support the claim that the use of computer generated TTS with a visual component can improve human perception. They also support the proposition that using both visual and aural channels of working memory can reduce the cognitive load and lead to better attentiveness and learning.

2.5.8 Summary

The use of audio, either musical sound or spoken audio, has not been implemented generally as animations in CS teaching. However, researchers have proposed it as a way to assess visualisation. Section 2.5 has shown that the use of audio in CS teaching is usually limited to basic forms using a computer's built-in speakers or, in more advanced implantation, as narrative voice. New audio based technologies tend to be more related to customer automated feedback tools. However, these technologies have great potential to engage students in interactive learning because the more realistic the feedback, the more that students will be encouraged to interact with the environment and, if it has a personality, the more that students will feel it is live interaction. There is, therefore, a good case for thinking that the use of aural instructions, together with visualisation, is an important approach for teaching the concepts of data structures.

2.6 Visualisation and Cognitive Load

Cognitive approaches to human ways of learning have highlighted the changes that occur with different mental representations of situations and tasks. To understand human cognition in cognitive architecture needs prior knowledge of models of human memory organisation, how the knowledge is represented and the problem solving mechanism (Kalyuga, 2006).

Kahneman (1973) and Navon (1984) explain that the concept of mental load is based on the concept of a communication channel with limited capacity. They argue that overloading this channel could result in missing signals and under-loading could result in a considerable spare processing capacity. The capacity theory of human processing in relation to attention was developed to explain a learner's limited capacity to perform many activities at the same time (Kalyuga, 2006). This means that when learning material is described distinctively in a basic form with low cognitive load, any irrelevant cognitive load caused by an extra attribute of the learning environment would have a little impact on the working memory. So, when designing instructional learning materials, cognitive load should be kept to its minimum.

2.6.1 Reducing cognitive load

Program visualisation had helped in reducing the cognitive load and releasing human processing capacity and reasoning, especially when graphics and visual aids are used to give learners extra insight and awareness of high level software programming (Tudoreanu, 2003). Tudoreanu produced guidelines, based on Sweller's (1994) CLT, to reduce the cognitive load in software visualisation. These included removing any indirect representation so that learners themselves can control how the visualisations are shown and can directly manipulate the produced visualisations. Then the visualisations should automatically determine the values of the graphical attributes of the objects shown and relieve the user from the need to define the functions and calculate the graphical values. Finally, he suggested enriching the visualization with an explicit representation of the visualization syntax and allowing the user to continuously adjust the appearance of the program views.

Although the use of visualisation can help in reducing the cognitive load, it would be of little use if packaged with an environment that consistently increases that load. In this case, the environment itself would cause the visualisation to have no impact on learners' understanding of the subject (Tudoreanu, 2003). Figure 2.12 shows Tudoreanu's illustration of two mental models. In the first, cognitive capacity is enhanced by using visualizations, but cognitive resources are diverted from understanding the program towards handling indirect structures that control how the program is represented. In the second illustration, the running program can be observed through visualizations that are customised directly via user interactions, and this leaves most of the cognitive resources available for solving other program related problems.

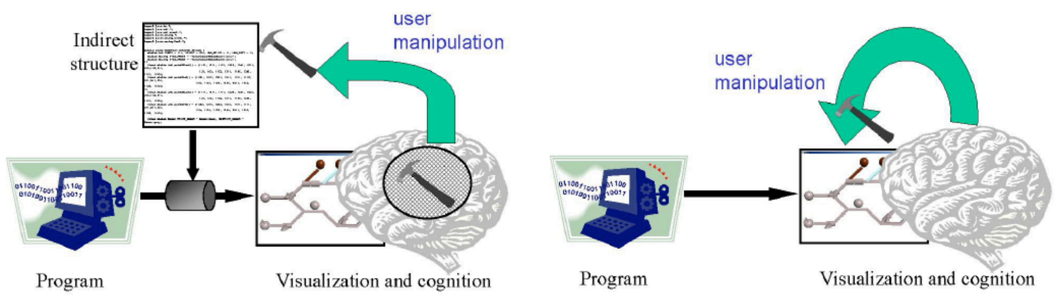


Figure 2.12: Tudoreanu illustration of how cognitive load affecting user learning experience

Mayer's (2001) model of information processing in multimedia learning was summarised by Ando (2008, p: 1). He defined multimedia as "a method for simultaneously presenting visual content (text, pictures, video) and narration (audio content)" and highlighted the fact that multimedia material can reduce the extraneous cognitive load on the learner. As a result of this insight, the 'dual channel model' has become an effective theoretical foundation for the use of multimedia materials in learning (Ando, 2008). Figure 2.13 shows Mayer's model of multimedia learning.

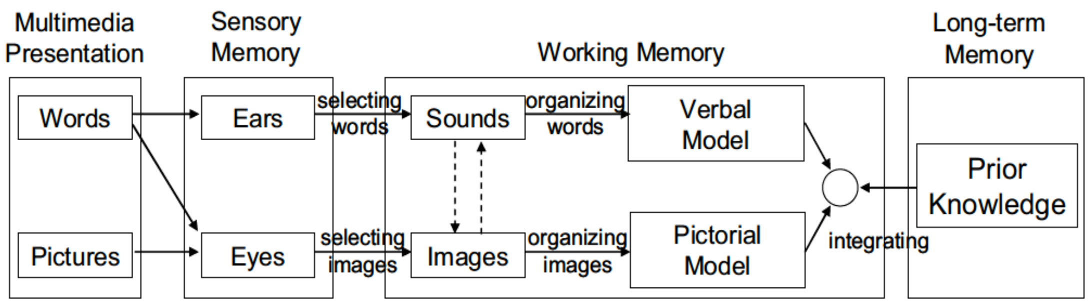


Figure 2.13: Mayer's multimedia learning model

Mayer’s (2002) Cognitive Theory of Multimedia Learning was based on three assumptions, as discussed in the following three sub-sections.

2.6.1.1 The Dual Channel Assumption

The dual channel assumption states that humans process information in two separate channels, visual and aural. Figure 2.13 shows Mayer’s (2002) model of the three stages of information processing where two types of information are presented to the learner. First, the information is acquired by the sensory memory by either listening or seeing. In the next step, information is processed separately but in parallel in the working memory where the memory tries to match and organize related images and sound together. Finally, the information can be integrated and linked to previously known information in the long term memory.

2.6.1.2 Limited Capacity Assumption

The limited capacity assumption states that learners have a limited information processing capacity for each channel at any given time. This means that the working memory can hold only a small number or images and sounds. Research in this field reveals that the working memory can process about seven pieces of information at a time, plus or minus two (Kalyuga, 2006). This suggests that, in an instructional presentation or learning environment, learners should be exposed to a limited amount of information when learning new subjects.

2.6.1.3 Active Processing Assumption

The active processing assumption suggests that learners can be actively engaged with the learning environment by selecting, linking and integrating the information they acquire with their prior knowledge. This can help learners in creating an effective mental model of the information they have just received (Hanley, 2010).

2.6.2 Split attention effect

The split attention effect takes place when instructional materials require learners to split their attention between multiple sources of information. Mayer's (2001) research concluded that the use of animation and associated narration are most effective when presented simultaneously rather than serially. This integration has been used for years in children's comic books and has proven its efficiency as most of the reading materials are cognitively demanding for children (Kalyuga, 2006).

According to the CLT, split attention can happen when learners try to integrate two associated sources of information. This attempt at integration might overload the limited capacity of the working memory when the learner tries to focus on both reading the text and looking at the images. However, overloading can be avoided if the working memory is enhanced by the dual mode presentation of the information. In conclusion, using aural narration can increase the capacity of the working memory (Sweller, 2002).

Gerjets and Kirschner (2009, p.257) have highlighted an important theoretical issue that challenges the currently dominant cognitive theories of instructional design for multimedia learning by Mayer and Sweller. They argue that those theories focus on the learner's cognitive system architecture, its limitations and ways of working with those limitations to design an effective learning environment. Pointing out that Mayer and Sweller came to their conclusions using experiments under typical controlled laboratory conditions, Gerjets and Kirschner (2009) argue that, not only might Mayer and Sweller's ideas not apply in actual learning environments but their application might reduce learning.

2.6.3 Response time and cognitive load

Hensler (2006) investigated the factors that influence students' performance in answering multiple choice questions. He developed an intelligent tutor model where the questions were generated by a speech automated speech engine. He concludes that performance can be measured by the time a student takes to respond to the questions and the difficulty of the questions themselves. He also reports that when the cognitive load is greater the level of performance is reduced.

This method of investigation was also used by Beck (2005), who explored how student engagement with intelligent tutors relates to productivity of learning. His model was based on "item response theory" where the correct results of a task are measured by the delay of interest rather than by the results themselves. Beck's model was based on the difficulty of the question and how long the student took to respond as well as whether the response was correct. His analysis of learner response times and correct responses suggested that students experienced active learning if they were engaged with the intelligent tutor. The lack of active engagement with task could also be an indication of higher cognitive load.

Dror (2005) studied whether older (mean age of 70) or younger (mean age of 18) adults adopt mental representations and processes that lower cognitive load and whether age affected response times. Participants were asked to mentally rotate a variety of images with different complexity levels as quickly as possible, maintaining correct answers to the tasks. The results showed that the response times of the younger adults increased as image complexity increased. However, for older adults the response time was the same. Dror concluded that increasing image complexity caused increasing cognitive load.

Khawaja et al. (2007) also investigated the current methods of measuring cognitive load. They presented users with two speech-based tasks of differing complexity and recorded the delay in responding, and the length and frequency of pauses. The differences in response time between the two answers were t-tested and this showed that the response times for the easier task were statistically significantly lower than that for the more difficult task. They concluded that higher levels of cognitive load resulted in increased response times.

2.6.4 Summary

The study of cognitive load is important in understanding how students gain knowledge. Cognitive load is the amount of mental effort needed to learn something; the greater the cognitive load, the more mental effort needed. So, designers of instructional learning materials should aim to keep cognitive load to a minimum. CLT is a model for instructional design based on the knowledge of how learners acquire, process and retain new information. The use of multimedia material can reduce the cognitive load on the learner's working memory. If the dual channel assumption is true, humans process information in two separate channels, visual and aural, so presenting knowledge that uses both channels can reduce the cognitive load. Learner's response time can be used as a measurement of cognitive load.

2.7. Chapter Summary

This chapter looked at the existing interactive teaching tools that enhance CS students' learning experiences and the research that led to their creation. In particular, it looked at how new computer technologies enhance these tools, especially those created for computer science students studying algorithms and data structures. This chapter also looked in some detail at learning styles and explained the importance of considering this concept when developing a teaching tool.

Prior studies have proved the ability of visualisations, discussed in Section 2.4, to facilitate the learning process, especially in CS education (Brown, 1991; Shilling and Stasco, 1992; Culwin, Adeboye and Campbell, 2006) and Section 2.4.3 provided details about a variety of visualisation tools. Analysis of these tools identified some design and scope problems and limitations. Most were designed as small programs and applets to investigate a simple programming concept, or they are focused on large-scale learning environments and could increase cognitive loads. Most of the tools were also evaluated in one short programming session followed by a student questionnaire. The target scope of previous studies varied among different age groups and they were not particularly aimed at first year CS students studying the concepts of data structures.

Section 2.5 moved on to look at audio use and research. The conclusion that can be drawn from that discussion is that, although auralisation is considered by many (e.g., Brown, 1992; Higgins, 2000; Vickers and Alty, 2005) to be beneficial to the learning process, insufficient work has been done to thoroughly investigate its importance in CS learning. However, Wolf's (2002) iWeaver system that used pre-recorded descriptions of programming subjects is probably most worthy of follow-up work. Other research investigated either visualisation only of programming concepts, or the use of audio with visualisations in learning tools not concerned with programming.

Section 2.6 discussion of cognitive load demonstrates the importance of considering its issues when designing an interactive visualisation tool, or any other type of learning aid or method. Poor interface design is likely to affect not only the learning process but also students' motivation and engagement in their learning process.

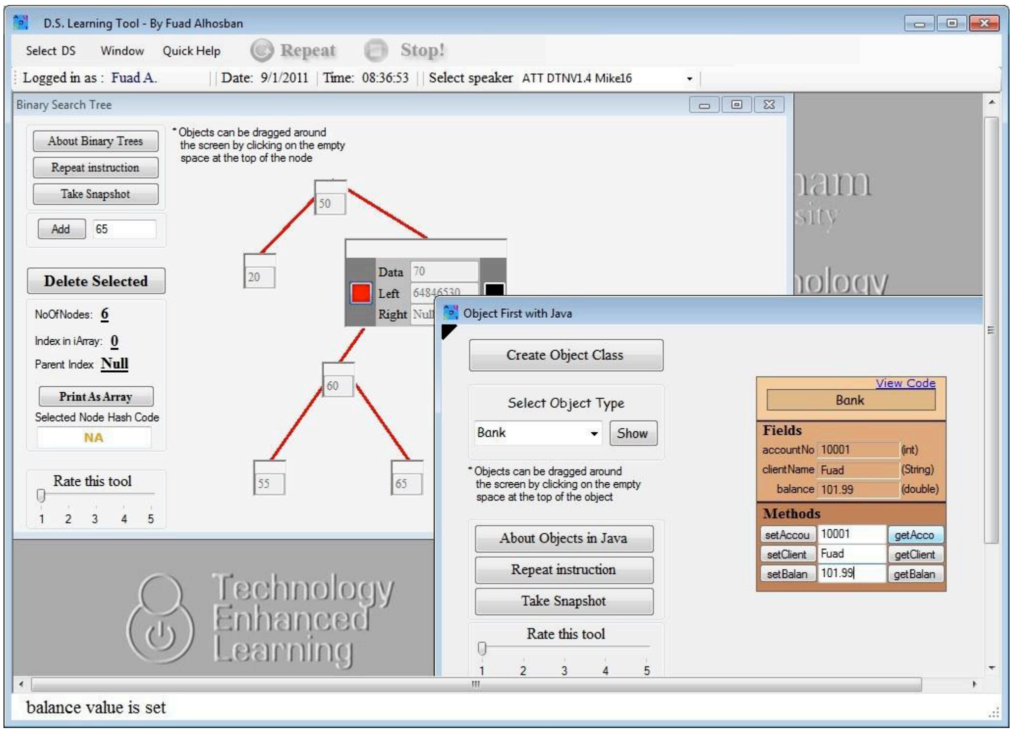
This chapter has examined previous studies that introduced new approaches to learning using interactive multimedia environment and, more specifically, those that investigated the use of e-learning tools to support teaching and learning of CS concepts by using visual, textual and aural support. Although several studies investigated the use of interactive multimedia tools to teach the concepts of data structures in computer science education, none has explored the use of aural instructions with visualisation to produce a focused approach with the aim achieving the lowest possible cognitive load by using Mayer's (2002) multimedia learning model. Also revealed was an absence of longitudinal research on the impacts of the combination of visual and aural displays, especially as a means to learn the concepts of data structures within the CS curriculum. That is the main focus of this research.

Chapter 3 : Implementation

3.1 Introduction

Chapter 2 reviewed many previous attempts to create software visualisation or tutoring systems that aim to help students in learning programming and data structures by creating either adaptive learning environments or interactive visualisation (Kazi et al., 2000; Cross, 2004; Karavirta, 2004; Bednarik, 2006; Culwin, 2006; Rajala, 2008; Woolf, 2008; Briana, 2009). The conclusion of that discussion in Chapter 2 was that none of the previous attempts had studied the efficiency of using interactive aural instructions, generated using TTS engines, together with visualisation, with the aim of creating a learning environment that reduced the cognitive load for novice CS students learning the concepts of data structures. Moreover, previous research (Cross, 2004; Kölling, 2006; Kölling, 2008) was based on creating either a very simple prototype that was not useful outside the lab where it was developed or a huge application with IDEs that distracted students' attention from the learning by introducing complicated environments so the application did not benefit their learning.

This chapter describes the requirements and implementation details of a prototype tool (the DSL Tool) that was developed to evaluate the integration of aural instructions with visualisation in a learning environment, as shown in Figure 3.1. It will focus on the technical details of the tool's core components and functionality that visualise interactive DS concepts in easy to use sub-programs called tasks, as well as integrating TTS technology to provide spoken feedback to students. First, the system architecture is illustrated. Then, Section 3.5 discusses the visualisation of OOP concepts in a non-object oriented structure developed in the Visual Basic (VB) Net programming language. It will also discuss the integration of speech, generated using AT&T Labs latest TTS engines, to produce high quality synthesised speech. The word "visualisation" in this thesis refers to the presentation of data structure in a visual form as on screen objects. However, it does not include animation of objects. Finally, Section 3.6 will explain the technical method for animating the visual components in the VB2005 development environment and how the interaction takes place.


D.S. Learning Tool - By Fuad Alhosban

Logged in as : Fuad A. | Date: 9/1/2011 | Time: 08:36:53 | Select speaker ATT DTNV1.4 Mike16

Binary Search Tree

* Objects can be dragged around the screen by clicking on the empty space at the top of the node

NoOfNodes: 6

Index in iArray: 0

Parent Index Null

Selected Node Hash Code NA

Data 70

Left 64846530

Right Null

Object First with Java

Select Object Type: Bank

* Objects can be dragged around the screen by clicking on the empty space at the top of the object

Bank		
Fields		
accountNo	10001	(int)
clientName	Fuad	(String)
balance	101.99	(double)
Methods		
setAccou	10001	getAcco
setClient	Fuad	getClient
setBalan	101.99	getBalan

balance value is set

Figure 3.1: Snapshot of DSL tool

3.2 The DSL tool's requirements

3.2.1 Requirements overview

Based upon the previous literature, this research adopted six important criteria, outlined below, for creating an effective learning environment to help first year CS students when they are learning the concepts of data structures. This research also presents the use of a new methodology to develop a focused learning tool that is an effective help to students and that can be used outside the experiment's boundaries. In addition, it considers the design of a prototype tool to be used for testing the effectiveness of using aural instruction along with visualisations which reduce the possible cognitive load in the learning environment.

The key requirements of a focused learning method to help students studying DS are:

- Aligning the DSL approach domain with the PDS module requirement.
- The DSL tool should be able to capture, organise and retrieve all the students' learning events.
- Narrative text or audio should accompany the visualisation, and the student should have the choice of using audio, text, or both feedback methods.
- Every visual component should be represented as a small object so that as many objects as possible will fit on a student's screen. However, it should allow the user to enlarge objects to display any extra information related to it. This will maintain a reduced cognitive load.
- The visualisations should be interactive and accommodate a student's actions.
- Students should be able to access and implement the DSL tool at any time within the computer science labs.

By analysing the efficiency of using aural instruction and the impact of using this method to enhance students' learning experience, it is expected that these elements will help this study to answer the research questions listed in Table 1.1.

3.2.2 Justifications of requirements

In order to provide validity to the DSL approach, the DSL tool's requirements are justified in this subsection, together with a brief discussion on how those requirements are related to the research objectives.

3.2.2.1 Aligning the DSL approach domain with the PDS module requirement

The first important issue that needs to be addressed when developing a learning approach is identifying the target audience. Knowing the users will determine the components that need to be investigated, as well as the breadth and the depth of the information that needs to be presented (Khuri, 2001). For this research, the target audience is first year undergraduate students undertaking the PDS module. So the domain of the DSL approach should be aligned with that module's requirements and it should focus on those components that the lecturer thinks students are likely to find most difficult to understand.

Rogers (2008, p.291) suggests a strategy that can be used when designing animations for learners. He argues that a good approach is to design simple visualisations that deliver particular components of a system rather than designing a complete set of visualisations that represent all the components of the system at the same time (Rogers, 2008, p.291). This approach is likely to result in less cognitive load on students' working memory when they try to relate multiple changes on the computer screen to what they are learning. In this case, students' focus may be on visual components that are not very important and they might focus less on the important subjects that relate directly to the modules' intended learning outcomes (Lowe, 1999).

3.2.2.2 The DSL tool should be able to capture, organise and retrieve all the students' learning events

In this research, there are two reasons why it is important to monitor student engagement with the DSL environment. The main reason for recording student usage is that the usage data will help in identifying how the student engages with the tool, and what learning events take place. The collected data on users' behaviour will help in analysing the effectiveness of the DSL methodology and answer questions related to what activities promote student engagement (Dawson et. al, 2008).

Recording usage data can also assist students themselves to organise their learning activities, and to see the knowledge they have acquired, and the visual components they have used at different times, so that they can integrate their learning experiences to create personal, meaningful records of their learning over the academic year (Vavoula, 2009). The recorded data can also provide concise and detailed user profiles that provide personalised learning targeted to students based on their attention given to certain learning objects (Najjar, 2006).

3.2.2.3 Instructional text or audio should accompany the visualisation, and the student should have the choice of using audio, text, or both feedback methods

This research is studying the impact of using aural instructions and feedback to accompany visual representation of the proposed data structures. To evaluate the effectiveness of audio with visualisation, the research also aims to evaluate the effectiveness of textual instructions with visualisation. In this way, the two instructional formats can be compared. Khuri (2001) emphasises the necessity of supporting visual representations with explanatory cues in the form of short notes, as they can increase the amount of information that the user receives through different working memory channels.

Mayer (2002, p.117) summarises a successful method of promoting understanding when he refers to “important aids to multimedia learning, in which students understand more deeply when they receive words and pictures.” Mayer (2002) reports that contiguous aids can help students in deeply understanding study materials if words and pictures are presented simultaneously on screen. He added that a modality aid can help students to the same extent if words are presented aurally rather than textually. Finally, the redundancy aid can help students in the same way if either audio or textual information is presented with visualisation rather than using them both at the same time. This assumption will be assessed in chapter 4 where the effectiveness of aural vs. textual narration will be evaluated based on students’ response times to instructions.

3.3 DSL Components' Design

This section describes the attributes of each of the DSL tool's components. It is important to describe each of the proposed components individually because this will contribute to understanding how the integration of aural instructions and visualisation works overall, and illustrate the DSL tool's features. Section 3.5 of this chapter will describe in detail the implementation of the tool's components.

3.3.1 Basic Objects

The objects refer to the simulation of the object in the first task that the students perform. This concept was presented to the students at the beginning of the term. It helped students in understanding what is meant by objects in Java, and how they are created. The design of this object simulates BlueJ's presentation of Java classes in order to build a relation with the programming environment that students will be working with during the academic year.

The created object can be dynamically changed according to the student's specification of its attributes. When creating an object, a student can name the object class. Based on that, an object is created that simulates the run-time state of the Java class. After that, the student can interact with the object by setting and getting its values.

3.3.2 Nodes

In data structures, nodes are referred to as the data record in the computer memory that forms the basic form of data structures such as Linked Lists, Binary Trees.

Nodes are an important component of the DSL design. They are used to simulate how DS actually work and are stored in the memory.

3.3.3 Integration of aural instructions in a visualisation

As aural instruction is a key factor in the DSL approach, it is important to present how the audio component is used as an instructional method. The use of speech technology allows the integration of aural instructions and the rendering of any text as spoken audio. The voice used in this research was chosen because it sounds the most natural of the voices available and it was the favourite voice of both staff and students.

3.4 Use case

The diagram in Figure 3.2 illustrates a use-case of the DSL tool. First, the student logs in to the tool to supply and verify his or her user information. Then, the student chooses his or her desired format for the instruction. The system is then set to deliver textual or aural instructions while the student is learning any of the data structures approaches. In the first mini study, students had no choice over the selection of instruction format, as they were grouped randomly in three groups, and each group had a specific format of instruction. However, when considering the results of the mini study, as explained in Chapter 4, students only had to choose either aural instruction or textual instructions.

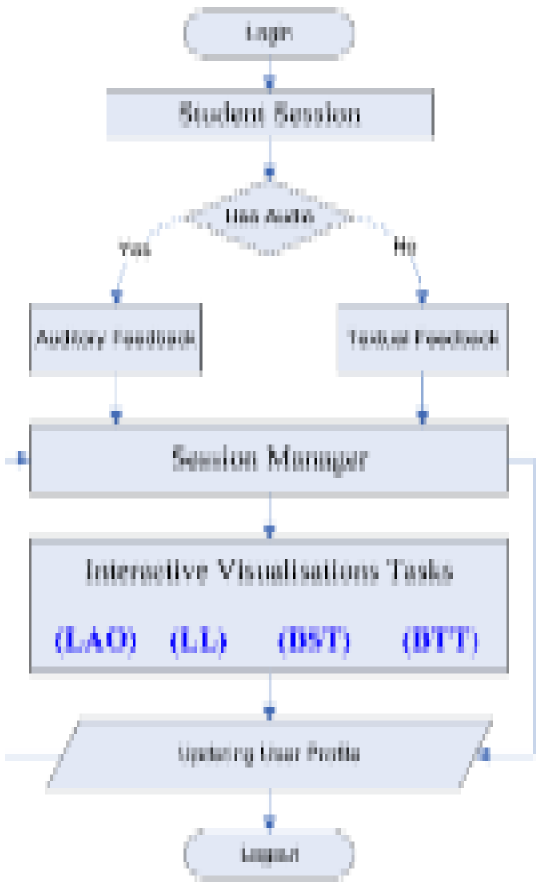
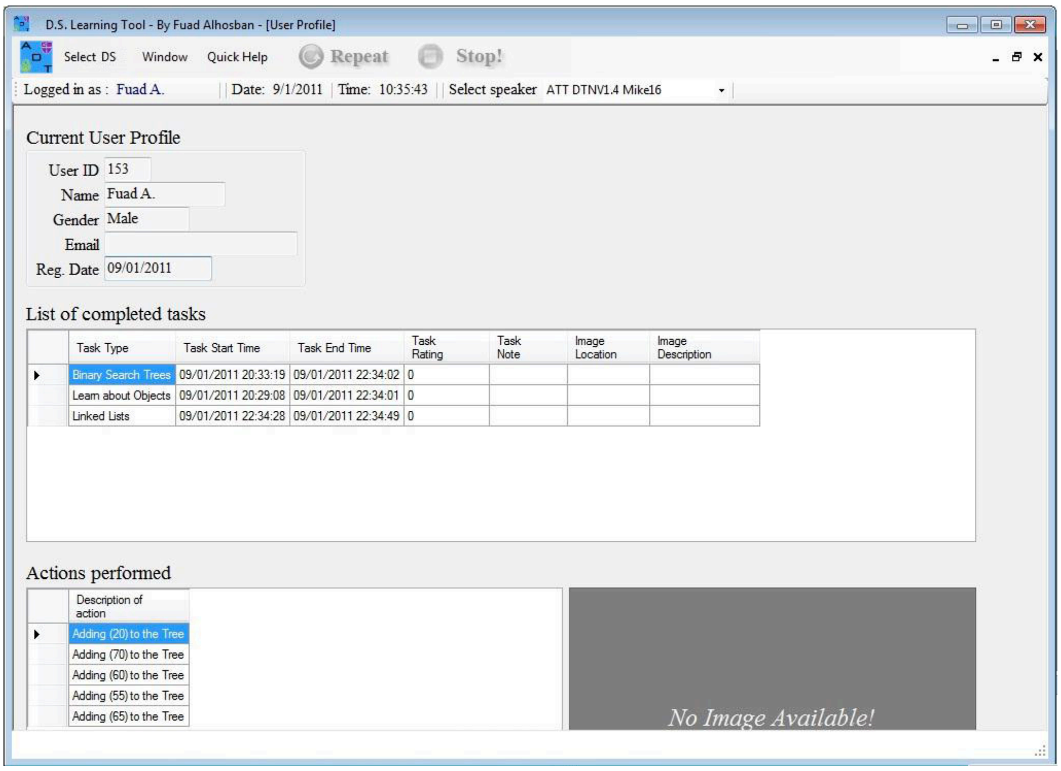


Figure 3.2: DSL Tool Use Case Diagram

The student can then start any of the visualisation tasks available in the tool (Learn About Objects, Linked List, Binary Search Tree and Binary Tree Traversal). During each task session, the student is asked to leave feedback about the task itself by either giving a rating (out of 5), or giving written feedback about his or her experience. When the student has finished working on a visualisation task, a snapshot of the student screen is taken and saved for later reference. At the end of the task session, all the recorded information about the student's usage is stored in the tool's database. Later, the student can review his or her usage data by accessing the User Profile screen shown in Figure 3.3.



Logged in as : Fuad A. | Date: 9/1/2011 | Time: 10:35:43 | Select speaker ATT DTNV1.4 Mike16

Current User Profile

User ID	153
Name	Fuad A.
Gender	Male
Email	
Reg. Date	09/01/2011

List of completed tasks

Task Type	Task Start Time	Task End Time	Task Rating	Task Note	Image Location	Image Description
Binary Search Trees	09/01/2011 20:33:19	09/01/2011 22:34:02	0			
Learn about Objects	09/01/2011 20:29:08	09/01/2011 22:34:01	0			
Linked Lists	09/01/2011 22:34:28	09/01/2011 22:34:49	0			

Actions performed

Description of action
Adding (20) to the Tree
Adding (70) to the Tree
Adding (60) to the Tree
Adding (55) to the Tree
Adding (65) to the Tree

No Image Available!

Figure 3.3: A snapshot of User Profile

3.5 Implementation of the DSL tool components

Section 3.3 presented an overview of the three main components of the DSL tool, namely, basic objects, nodes and embedded TTS engines, and their proposed functionality. This section describes the technical attributes of each of the three components. The notion of nodes, linked lists and binary search trees in this thesis are referred to as parts of data structures.

3.5.1 Basic objects

As stated above, objects refer to the simulation of the object in the first learn about object task that the students perform. This evaluation of this task will be discussed in detail in Chapter 4. The object component was created in the Visual Basic VB 2005 programming language as a “User Control” component, that is, a sub program that can interact with by a single object (e.g. button or text box).

Figure 3.4 shows the design of the basic object in both “design-time” and “run-time” state. Design-time state is the appearance of the object in the programming environment before running the tool, while the run-time state is the appearance of the object during the run of the tool.

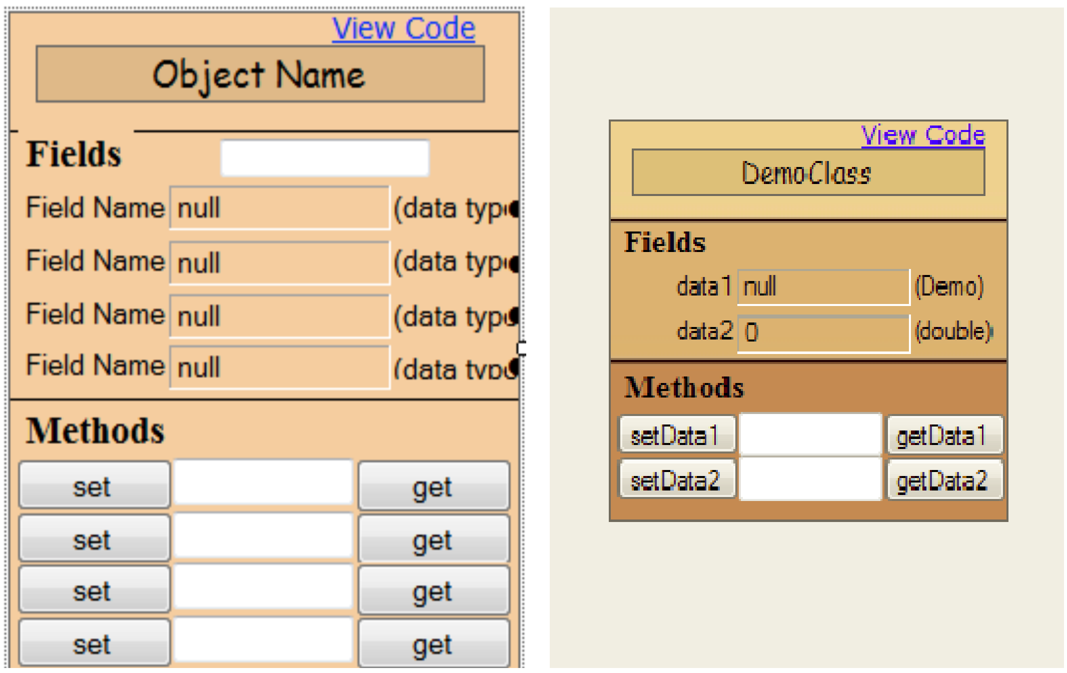


Figure 3.4: Design-time and Run-time states of basic objects in DSL Tool

When creating an object, a student can name the object class (e.g. Shape, Bank Account, Student … etc.) and specify how many fields he or she wants to have and the data type associated with those fields (e.g. data1, name, balance … etc.). The student can then set or get the data stored in those fields by pressing “setData1” button or “getData1” button to simulate Accessor and Mutator methods in Java. The student can also have access to the code used to generate the object shown, and this code can be used directly in any Java program.

3.5.2 Nodes

As explained in Section 3.4.2, a node is an important component of data structures so it is an important component to be visualised. Nodes are used to simulate how data structures actually work and they are stored in the memory.

Similar to basic objects, nodes are also a “User Control” component created in VB2005. The DSL tool can control the nodes in the same manner as in basic objects by interacting with them. Figure 3.5A presents two different states of the Linked List node component, and Figure 3.5B presents two different states of Binary Tree node.

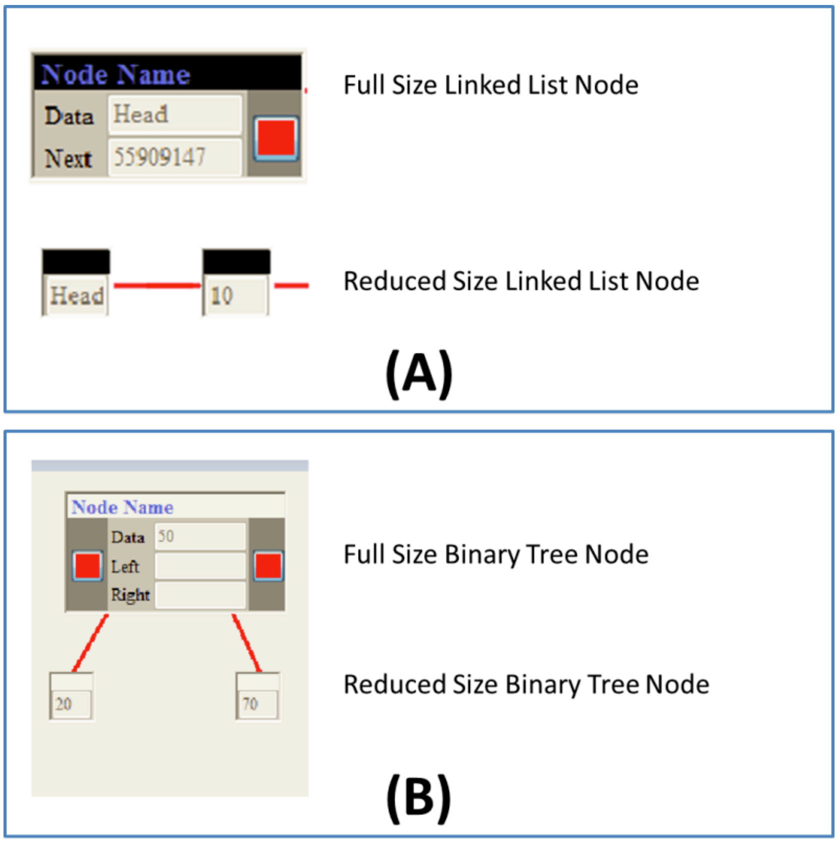

Node Name
Data
Head
Next
55909147
Full Size Linked List Node
Head
10
Reduced Size Linked List Node
(A)
Node Name
Data
50
Left
Right
20
70
Full Size Binary Tree Node
Reduced Size Binary Tree Node
(B)

Figure 3.5: (A) Linked List Node (B) Binary Tree Node

Figure 3.5 illustrates two types of nodes used in the DSL tool in their run-time state. Figure 3.5A shows the full size and the reduced size of the Linked List node that contains both the value stored in the node and a single Hash Code (address location in memory) of the next node connected to it. The red square box shows that the node is linked to another node, but a black box is shown if the node is not connected to any other node (leaf node). Figure 3.5B illustrates a Binary Tree node that works as a Linked List node; however, it holds the left and right memory location of its children.

3.5.3 Embedded TTS Engine

As explained in Section 3.3.3, the availability of interactive aural instructions is a key factor to the DSL environment. It is important to show how the audio generation component was added to the tool and how it is controlled. To use TTS in the VB programming environment, the Microsoft Speech Library that controls any installed TTS on the system was linked to the DSL tool. The speech library allows the system to synthesise any text passed as a parameter and convert it to spoken audio. The voice used in the prototype tool was developed by AT&T Labs Natural Voices®. The voice of a woman (Audrey) with a British accent was used as it was favoured by both students and staff. Although other voices are available, only the Audrey voice was used to reduce the number of experimental variables that could affect the results.

The code snippet shown in Figure 3.6 summarises how to programme the TTS engine. The code was used in the VB development environment to allow the DSL tool to synthesise text in each visualisation task.

```
Private sayText As String
Private WithEvents SayThis As New SpeechLib.SpVoice
SayThis.Voice = MDI.MDIVoice.Voice
sayText = "You can now start creating objects by clicking on 'Create Object Class'"
SayThis.Speak(sayText,SpeechLib.SpeechVoiceSpeakFlags.SVSFlagsAsync)

Private Sub SpeechThread_DoWork(ByVal sender As System.Object, ByVal e As

    System.ComponentModel.DoWorkEventArgs) Handles objectsThread.DoWork

    SayThis.Speak(sayText,

    SpeechLib.SpeechVoiceSpeakFlags.SVSFlagsAsync)

End Sub
```

Figure 33.6: Summary of coding the TTS engine in VB

As shown in Figure 3.6, to convert the text to speech, first an object called "SayThis" is created along with a string variable to hold the text to be synthesised. Then, the created object is linked to a voice object created within the DSL environment (MDI.MDIVoice.Voice). Finally, the text is passed to the SpeechThread method for conversion. The thread function is used because of the delay in the system caused by halting the execution of the tool until the speech stops. The threading function allows the system to process the speech in parallel with the main program.

3.6 Animating the visual components

To design an interactive learning tool that can interact with the visual components of the DSL approach, it was important to produce simplified visualisations to allow the students to manipulate and move objects around the screen. The interaction with objects happens only after they are created by students. In the Learn About Objects task for example, the created objects had the capability to be dragged around the student screen. Moreover, it is important to note the capability of the tool to visualise inheritance (when an object has a field of another object type). To achieve that, a link is automatically drawn on the screen to show the inheritance relation between the two objects. This is a similar approach to that adopted by the BlueJ IDE, and so learners are likely to be familiar with this approach.

In Linked Lists tasks, two methods are used to build an interactive list structure. The first method creates the list structure in an object oriented manner, as it is created in Java. The tree structure is easily represented in the memory because VB supports OOP. However, to produce the animation, a more complicated workaround has to be implemented. Two array lists are created to hold down the information about the node values and their locations on screen and in the memory. Then, a new function is created to loop through all the lists and objects shown on the screen to match their values and maintain consistency between the visual part and the actual values stored in the memory. Whenever a node value is changed or dragged around the screen, the function checks again for consistency between the values.

In Binary Search Trees and Tree Traversal tasks, the same methods are used to store the nodes location on the screen and their values in two array lists. However, more complicated algorithms are used to simulate the deletion of nodes from the trees as the deletion can result in restructuring the whole tree and result in many nodes being relocated in different locations on screen depending on what node was removed.

Chapter 4 : Preliminary Study

4.1 Introduction

The discussion, in Section 2.6.3 of this thesis, about the relationship between response time and cognitive load suggested an answer for the first research question “What existing research evidence is there that the simultaneous use of aural instruction along with visualisation will reduce cognitive load when learning data structures?” It was shown that there is evidence (Beck, 2005, Dror, 2005, Hensler, 2006, Khawaja, 2007) that supports the use of response time as a measurement of cognitive load reduction.

This chapter focuses upon the pilot experiment which was conducted to ensure that the planned research methods would help in answering the second research question “Does the combination of aural instruction and visualisation reduce students’ response time for task completion compared with textual instruction and visualisation?” and prove the validity of the research hypothesis H1 “Reducing cognitive load improves student engagement and outcomes when learning data structures.” The pilot study also aims test the functionality and reliability of the research environment to set the correct path to answer the rest of research questions identified in this research.

The design of the DSL environment aimed to assess the effectiveness of using textual and aural instructions in a visualised Computer Science (CS) learning environment. This experimental study was designed to reveal how on-screen instructions can benefit CS students in learning the concepts of data structures. More specifically, it measures if CS students' response time to a given aural or textual instructions is reduced, thus implying reduced cognitive load. The results obtained from this study will answer Q2, which is "Does the combination of aural instruction and visualisation reduce students' response time for task completion compared with textual instruction and visualisation?"

The experiment was conducted during the first practical session of the Introduction to Programming module when most CS students had little or no previous contact with computer programming. Feedback given by the students about the learning environment, and their reports of any technical issues with the DSL prototype tool was recorded so that it could be assessed and used for future recommendations and updates of the tool.

4.2 Study design

In this mini study, first year undergraduates in the CS department at Durham University used the DSL environment as an introduction to learning about objects. To do this, they used a prototype version of the DSL tool to be used to test the validity of the research hypotheses. This was, for some students, their first real exposure to the concept of objects and the use of Java programming. Students doing the PDS modules were asked to voluntarily complete a task to help them understand how objects are built in Java, and the BlueJ environment was used for this purpose. The DSL tool monitored students' usage and collected data about their task completion.

Task completion is measured by the student's ability to produce a visual representation of the created objects. Figure 4.1 shows an example of a complete task where the student created a visual representation of the Bank object.

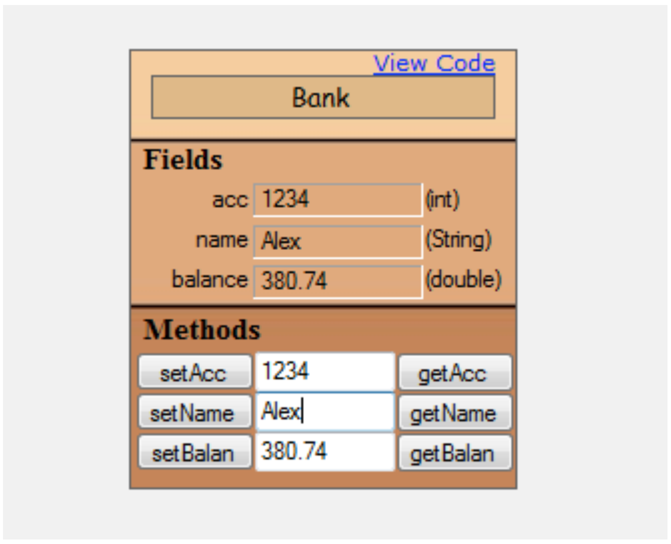


Figure 4.1: Visual representation of Bank object created by DSL tool

4.2.1 Study Subjects

A total number of 30 students were involved in this experiment. The experimental procedure applied was based on dividing the students into three conditions in a between-subjects design. In the first and second conditions, at the beginning of the practical session, each student in the two groups was given a laptop and a headset. In the third condition, the remaining group used the non-audio version of the prototype tool, so they were given only laptops.

The subjects for each group were distributed randomly. The process of allocating conditions to students was based on randomly assigning laptops with different instruction formats of the tool to students during the PDS practical session. Based on that, each group contained variety of student behaviour, experience and learning style. However, the identified risk of using this method is that it can produce non-equivalent groups of students.

Group	No. of Students	Instructions Format
A	10	Textual
B	10	Aural
C	10	Textual and Aural

Table 4.1: Groups and Conditions table

4.2.2 Variables

In this study, the experiments' identified dependent variable is response time, that is, the time a student takes to respond to a set of instructions to complete a given task. The independent variables are the use of aural instructions only with visualisations, the use of textual instructions only with visualisations, and the use of aural and textual instructions together with visualisations.

To eliminate any outside variables that could affect the results, the conditions of the environment during the experiment were all kept the same. All the laptops used had the same performance level.

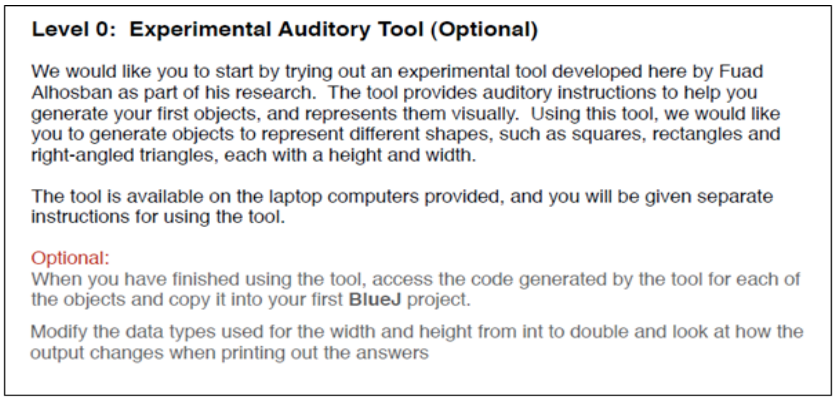
4.2.3 Subject variable confounds

There was a concern about the experimental subjects' variable confounds, that is that different students may behave differently during the task in a way that may affect the results of the experiment. The possible confounds that could affect the results of this mini study are: students' learning styles, differences in age among groups, language issues that impacted on understanding instructions, students' reading speeds, and prior knowledge of the subject. If any of these confounds correlated with the independent variables of the study and resulted in changing the value of the dependent variable, then the whole study would be confounded.

As explained in Section 4.2.1, a randomisation process was used to reduce the subject variable confound. However, there was still a risk in having non-equivalent groups of students.

4.2.4 Experiment of Procedure

The task (learning by using the DSL tool) was an optional part of the practical session. It was made clear to the students that they could start their practical without doing the task and that not taking part in the experiment would not affect their assessment. They were asked to participate if they thought it would be beneficial to further understand the concepts of objects in Java. From 44 students undertaking both IP and PDS modules, 30 students volunteered to take part in the experiment. Figure 4.2 shows an extract from the practical task sheet.


Level 0: Experimental Auditory Tool (Optional)

We would like you to start by trying out an experimental tool developed here by Fuad Alhosban as part of his research. The tool provides auditory instructions to help you generate your first objects, and represents them visually. Using this tool, we would like you to generate objects to represent different shapes, such as squares, rectangles and right-angled triangles, each with a height and width.

The tool is available on the laptop computers provided, and you will be given separate instructions for using the tool.

Optional:
When you have finished using the tool, access the code generated by the tool for each of the objects and copy it into your first **BlueJ** project.

Modify the data types used for the width and height from int to double and look at how the output changes when printing out the answers

Figure 4.2: Lab Class quotation

At the beginning of the task, students were asked to complete login details and some demographic information including, name, age, gender and programming experience. This information was kept in the system for later statistical and analytical use. The students also used their logging details when using the full version of the prototype tool. To ensure that the system was working correctly, the students were then asked to verify that they could hear/see a sample of a spoken or written instruction.

By the end of the task, students should have gained an idea about how objects are created and should have developed a preliminary understanding of the OOP concept. This was considered a good start for first year computer science students, as it enabled them to visualise the basics of programming languages (Cross and Hendrix, 2006). More importantly, the pilot study aimed to determine the best form of instruction in order to achieve the most effective results in the shortest time.

In this experiment, five main instructions of increasing complexity were monitored. The collected data were then grouped into tables based on the different types of instruction.

A number of procedural variable confounds that could affect the experiment procedure were identified; these variables related to the context of instruction presentation and the length of the instruction. To control these potential confounds, all the textual and aural instructions were identical, and the length of time that textual instruction showed on the screen was made exactly the same as the length of time that aural instructions took to play.

4.3 The DSL Environment

The objective of this phase of the research was to use the DSL tool to study the effect of using three methods of giving instructions to students in a visual interactive environment, that is, the DSL environment. It also aimed to answer the research question Q2: “Does the combination of aural instruction and visualisation reduce students’ response time for task completion compared with textual instruction and visualisation?” The dependent variable in this experiment is the reduction of cognitive load and its impact on learning, as measured by the response time to the instruction given.

The first method of instruction was using text based instruction only. This was considered a traditional way of giving instructions. It was also the simplest form of an interactive media experience, and suitable for learners who recall material by reading it. The second method presented the same textual instruction but it also used spoken audio instructions. The final method combined both textual and audio instructions at the same time.

In order to check a student’s response times, the tool first greeted the learner, directed the learner to a simple instruction to run the programme, and made the student aware of the availability of a repeat button that allowed the student to hear or see the instruction again. These first two instructions were not monitored by the tool, but they allowed the student to become familiar with it.

After that, the tool started to monitor the time between the instruction being given and the student responding by completing the task requested in the instruction. As explained in Section 4.2.4, to reduce procedural confounds that could affect the result, each instruction was given or shown for exactly the same length of time. For example, the duration of the instruction “Start a programme by clicking on Start button” was 3.35 seconds for both the aural instruction and the textual instruction. The text then disappears from the screen. Figure 4.3 shows an illustration of this example. An identified risk that could affect the results of using this method is the assumption that playing the aural instruction for the same length of time as showing the textual instruction has equivalent effects on a student.

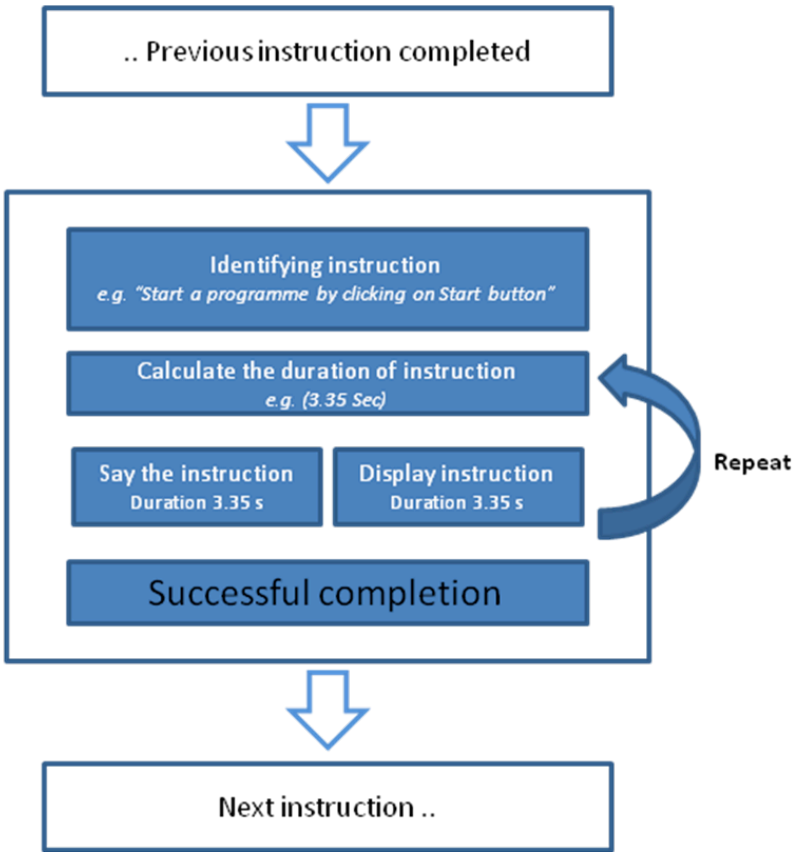


Figure 4.3: Illustration of instruction format example

4.4 Results

This section describes the collected quantitative data generated by the DSL tool based on students' usage. The dependent variable conditions were investigated in a between-subject analysis. The impacts of the three conditions on response time, the independent variable, were measured and recorded. The mean of each student's response time, in seconds, was then calculated and these calculations are summarised in Table 4.2.

Text Only Group	Audio Only Group	Text and Audio Group
10.40	9.40	10.40
10.20	8.80	9.20
12.40	8.60	9.20
13.40	7.80	6.00
13.40	9.80	11.00
18.40	10.60	13.60
8.60	10.00	18.00
6.20	8.80	10.00
14.80	8.80	9.00
16.60	10.00	12.80

Table 4.2: Means of Response Time in seconds for each student in different method groups

Further testing of the data was necessary in order to continue to answer the second research question, "Does the combination of aural instruction and visualisation reduce students' response time for task completion compared with textual instruction and visualisation?" The collected results were expected to show that the use of audio both with and without textual instructions would improve the learner's performance, show effective usage of the environment and, thus, demonstrate that cognitive load was reduced. In order to support the first hypothesis, H1, of this research that "Reducing cognitive load improves student engagement and outcomes when learning data structures", a test of significance was required. However, a normal t-test would not be valid for a comparison of three groups as it can only be used to compare means between two groups (Field, 2007, p.349). Consequently, the One-Way Analysis of Variance (ANOVA) was used because this allows for comparison of three or more experimental conditions and it provides details about the differences between these groups (Field, 2007, p.348).

The results of the ANOVA analysis were as follows:

Descriptives
Response Time in Seconds

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean		Minimum	Maximum
					Lower Bound	Upper Bound		
Audio Only	10	9.2400	.86307	.27293	8.6226	9.8574	7.80	10.6
Text and Audio	10	10.7600	2.73707	.86554	8.8020	12.7180	6.40	16.0
Text Only	10	12.6000	3.54526	1.12111	10.0639	15.1361	6.20	18.4
Total	30	10.8667	2.89986	.52944	9.7838	11.9495	6.20	18.4

Table 4.3: One-Way ANOVA test results

Table 4.3 shows that using audio only as the instructional method had the shortest mean response time and, thus, that audio instructions were more effective than the other two types. Text and audio instructions together produced the second shortest mean response time and the longest response time was for the text only instructions.

Table 4.4 shows the homogeneity results of the variance test. This test is designed to examine the null hypothesis that the variances of all three groups were the same; its results help to the answer the first research question. Next, Levene's test was used to check whether the different distributions of the data were statistically significant or if they could reasonably have come about by chance (Ellison, Barwick and Farrant, 2009, p.80).

ANOVA Response Time in Seconds					
	Sum of Squares	**df**	**Mean Square**	**F**	**Sig.**
Between Groups	56.619	2	28.309	4.082	.028
Within Groups	187.248	27	6.935		
Total	243.867	29			

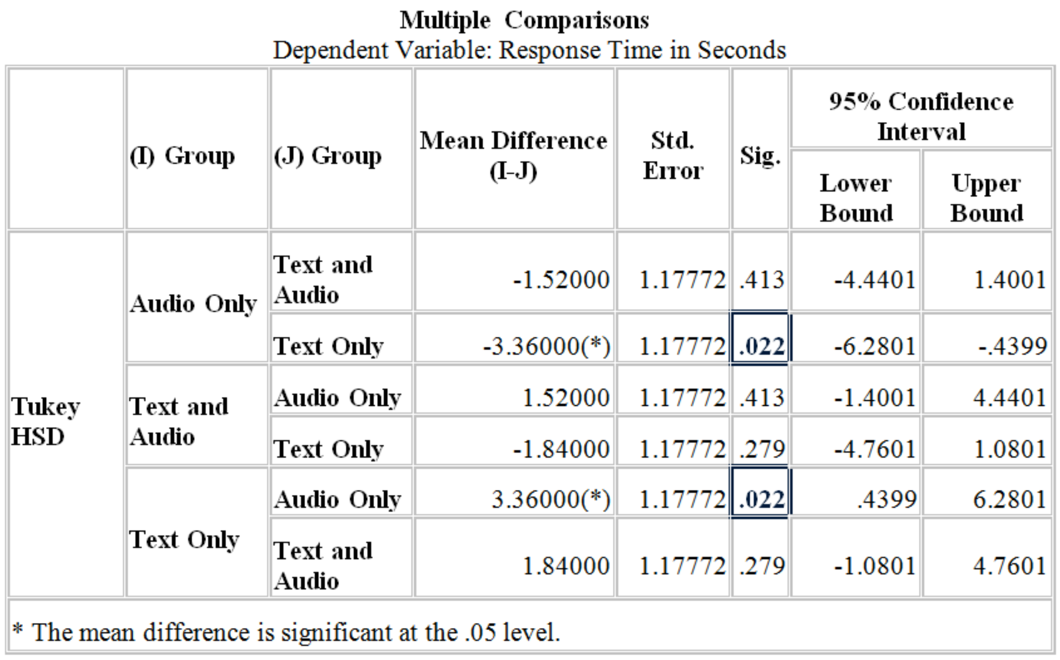
Test of Homogeneity of Variances Response Time			
Levene's Statistic	**df1**	**df2**	**Sig.**
4.316	2	27	.024

Table 4.4: Further ANOVA Results

Table 4.4 shows that Levene's test revealed that the variances were significantly different (the value of Sig <0.5). This means that the results violated one of the assumptions of ANOVA and that there were significant differences between the groups.

A multiple comparison test was then performed and the output gave the results required for the Post-Hoc test. Use was made of Tukey's Post-Hoc multiple comparisons, which compares each method with the other two, in pairs (Field, 2007).

The post-hoc results are shown in Table 4.5.



Multiple Comparisons

Dependent Variable: Response Time in Seconds

	(I) Group	(J) Group	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Tukey HSD	Audio Only	Text and Audio	-1.52000	1.17772	.413	-4.4401	1.4001
		Text Only	-3.36000(*)	1.17772	.022	-6.2801	-.4399
	Text and Audio	Audio Only	1.52000	1.17772	.413	-1.4001	4.4401
		Text Only	-1.84000	1.17772	.279	-4.7601	1.0801
	Text Only	Audio Only	3.36000(*)	1.17772	.022	.4399	6.2801
		Text and Audio	1.84000	1.17772	.279	-1.0801	4.7601

* The mean difference is significant at the .05 level.

Table 4.5: Multiple Comparisons Table

Table 4.5 shows that there is a statistically significant difference between the audio only and text only instructional methods. There is no statistically significant difference between the text only and text with audio instructional methods.

The overall plot of mean of the response times, in seconds, is shown in Figure 4.4. This provides a visual comparison of response times.

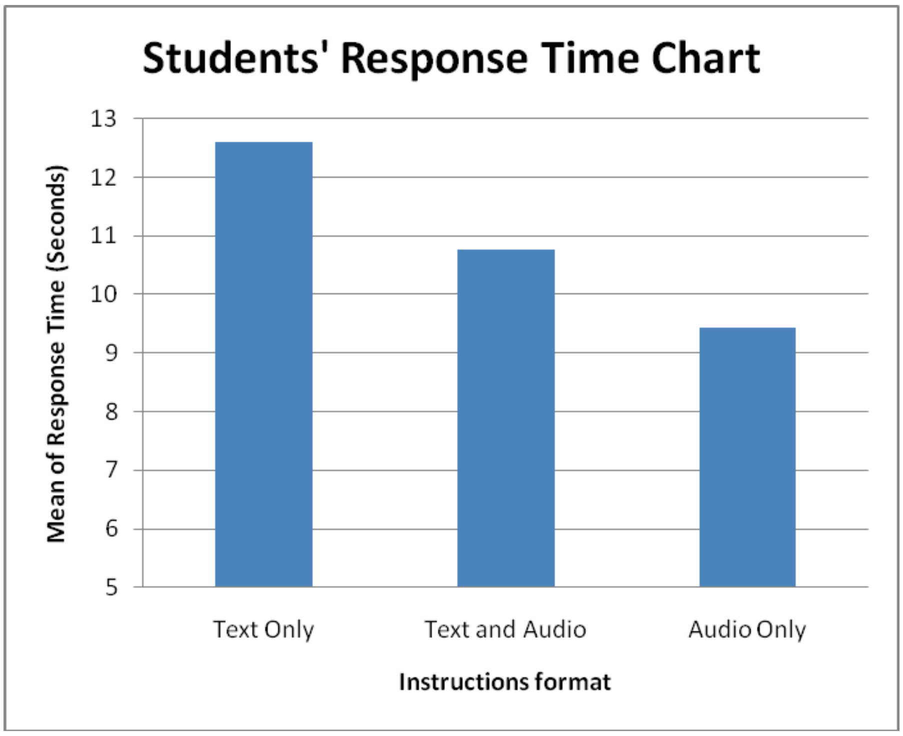


Figure 4.4: Mean of Response Time Plot in seconds

4.5 Discussion

As shown by the statistical analysis of the learners' response times, students who used the DSL environment responded significantly more quickly with a successful outcome to audio instructions than to other types of instruction. This leads to the expected conclusion that aural instructions do benefit students in a visual interactive learning environment. Those who used audio together with textual instruction performed better, but not significantly better, than those who used textual instruction only. The results show that there is a potential benefit for using aural instructions in visualised learning task. This benefit is measured by the reduction of the response time to on-screen instruction in successfully completing a learning task.

The tool usage data and how students behaved during the experiment was investigated further by looking at the students' use of the Repeat button. The students had been told that they could click on the Repeat button if they did not understand the instruction the first time they received it. The recorded data showed that the Repeat button was used only twice by the students who used audio but it was used four times by students who used both the text with audio and the text only instructions. To investigate the impact on the overall results of using the Repeat function, the records of students who used the Repeat button were removed and the data were examined again. The results showed that use of the Repeat function had no impact on the results found previously. Therefore, it can be concluded that the students still performed better with audio only instructions.

Investigating the second research question, Q2 "Does the combination of aural instruction and visualisation reduce students' response time for task completion compared with textual instruction and visualisation?", this mini study examined the results of using two types of instructions (aural and textual) in three experimental conditions. A comparison of response times in the three groups to instructions revealed that the use of aural instruction produced statistically significant lower response times than did the use of textual instruction. This result enables an affirmative answer to be given to the second research question.

4.6 Conclusion

The conclusion is that using aural instruction resulted in a significantly quicker response to instruction comparing with the use of textual only instructions. However, when using simultaneous aural and textual instructions, students' responses were not significantly quicker than when textual only instructions were used. This could be caused by students being distracted when using the dual instruction method, and this conclusion is supported by Kalyuga (2006) who found that using both text and audio instructions caused students to divide their attention between the two. The student's limited working memory capacity may be overloaded when he or she tries to focus on reading the text and looking at the images at the same time. However, cognitive overloading can be avoided if the working memory is enhanced by the dual mode presentation of the information, as discussed in Section 2.8.

The results of this study suggest that a strongly positive answer can be given to the second research question, Q2. "Does the combination of aural instruction and visualisation reduce students' response time for task completion compared with textual instruction and visualisation?" However, there is no evidence that the simultaneous use of textual and aural instruction can significantly decrease students' response time. These results, allied to the analysis made in Section 2.6.3 of the literature on cognitive load, supports the proposition that cognitive load can be reduced by the DSL approach.

In summary, using aural instruction can increase the capacity of the working memory. Based on this result, the research continued into its second part, using audio only instructions with visualisation and abandoning the use of text with audio instructions. It is expected that using audio only instructions will produce an improvement in students' interaction with the DSL approach.

Chapter 5 : Research Methods

Several research methods are used in this thesis. Studying the effectiveness of aural instructions with visualisation in the DSL environment depends mainly on collecting quantitative data, but some qualitative data also needs to be collected to support the findings based on the quantitative data. If the results obtained by using one method corroborate the results obtained by using the other method, triangulating these results adds strength to the answers to the research questions listed in Table 1.1.

Chapter 2 and chapter 3 presented an argument that supports the first research hypothesis H1, and answers the first and second research questions. This chapter presents the research methods used to address the remaining three research hypotheses and seven research questions presented in Table 1.1.

5.1 Study design

5.1.1 Overview of the study

As stated in Chapter 4, first year undergraduates in the CS Department at Durham University used the DSL environment to help them in understanding three concepts of data structures, namely, Linked Lists, Binary Search Trees, and Binary Trees Traversal. Students taking the PDS modules were introduced to the DSL tool and given brief training on how to use it in a practical session immediately after the PDS lecture introducing each data structure concept. Volunteers were asked to use the tool to help them understand how the selected data structures concepts work in Java, for example, in the BlueJ environment. The students had access to the DSL tool whenever they wished to use it. There were 30 volunteers but a total number of 27 participants engaged with the tool during the study period. The DSL tool monitored their usage and collected data about each visualisation task they completed.

5.1.2 Rejected research methods

The cross-sectional experimental method used in the first experiment was replaced by the use of longitudinal experimental method in the second and main experiment. The cross-sectional method was excluded from the second study for two reasons. The first was due to nature of the study itself, which tries to assist novice students over a whole academic year. It would not have been ethical to assist some students and not others over the length of an important assessed course. Secondly, the prototype tool's usage data was designed to be collected automatically by the tool itself, and the learners were expected to use the tool in different ways that are not amenable to the cross-sectional experimental approach.

In addition, a longitudinal design can achieve a more comprehensive evaluation of the design and help in investigating reasonable associations between the DSL environment and the effectiveness of a multimedia learning environment (Hu et al, 2007). The study was a conducted over two terms of the academic year, that is from 15 January to 25 June 2010.

5.1.3 Ethical Approval

Durham University requires every student to obtain ethical approval for his or her research design before starting any experiment. Oates (2006, p 55) explained that research committees will need to satisfy themselves that the subjects affected by the study will not be harmed, and that all the people involved will receive fair treatment. She listed the following issues that need to be addressed before ethical approval is given:

- Specific data protection rights of participating individuals.
- Whether it is possible to offer incentives to the participants (such as a prize draw) to encourage them to take part in the research.
- Intellectual property rights to the experiment's components (multimedia components).
- Restrictions on the type of learning technologies that are allowed to be used during experiments.
- The legal liability of the developer of the system to be designed.

The research proposals for this study were submitted to, and approved by, the Department of Computer Science at Durham University. The DSL tool was also reviewed by Durham University Computing and Information Services before it was installed on computers in the Data Structures module laboratory.

5.2 Quantitative data collection

Quantitative data analysis in this research is related to evidence based on finding patterns through various forms of numerical data. The types of quantitative data collected in this research included Nominal, Ordinal, Discrete and Continuous data types. The data sources varied from automatically generated usage data and usage rating by students, to students' marks and the results of a questionnaire.

Each of those sources is discussed below.

5.2.1 Automated collection of usage data

This method of data collection was designed to support the findings of both the qualitative and the quantitative data collection methods. The prototype tool that was developed to evaluate the DSL approach was created with functionality to record each student's actions while using the tool. The collected data can then be used to detect any pattern that can provide evidence about the tool's efficiency. Since students' usage was to be monitored over the whole of the second term, the resulting data was expected to provide rich evidence for analysing the research results, and to help greatly in answering the research questions.

The recorded usage data was in the form of:

- Logging information (user name, time and date)
- User choice of feedback type (aural or textual)
- User choice of task (data structures exercise)
- Task log in and out time
- User rating for each task
- User freeform feedback (qualitative data)
- Values added, deleted, manipulated or printed
- "Repeat Instruction" count
- Accessing user profile

And for the assessment part:

- Trials count
- Correct and wrong answer count

At the first engagement between students and DSL tool, some demographic data was collected about each student's gender, age, and expertise in programming. A snapshot of this form is shown in Figure 11.1 in the Appendix. This information would be used in the eventual analysis of the collected data. In addition, the tool recorded the response time as the interval between the system delivering an instruction and the learner responding to it. However, the response time was only recorded in the first mini study, not in the second study.

The automated data recording function would also enable the researcher to reconstruct each student session, to look at the task that the student performed and to compare the feedback received to what the student actually did. This information would increase the validity of the results of other data collection methods, and help in discarding any irrelevant results that might affect the overall finding of the research. It would also help in building case studies because it would enable the researcher to identify students who were actively engaged with the tool.

The automated data collected by the DSL tool was designed to help in answering the research questions listed below.

Q4: Is there a relationship between visualisation type and CS students' choice of instruction type?

> The DSL tool collected information about the student's choice of instruction type with each visualisation task. This data enabled a relation to be examined between the choice of visualisation task (Linked Lists, Binary Search Trees, or Tree Traversal) and the preferred type of instruction. In turn, this relation can provide evidence about the effect of using aural instructions on students' perception of some data structure concepts, and about any tendency to use aural instruction only for harder to understand concepts. Since the level of complexity differs between the three data structure tasks, some are harder to understand than others.

Q6: Do CS students choose to use the DSL tool while studying data structures?

> The auto-collected information would provide evidence to evaluate the claim that students would opt-in to use the DSL tool as a regular resource to help them in understanding the concepts of the selected data structures. The frequency of usage would provide evidence about whether students perceive the DSL environment as a positive benefit to their learning.

Q7: Of the three data structure types used in this study, which do CS students select to explore through the DSL tool?

> As mentioned previously, the collected data will provide information about the use of the DSL tool with each visualisation task. This information will provide evidence about the level of positive benefit perceived in each task, and about the way in which the DSL environment helped the learners.

5.2.2 Questionnaire

A research questionnaire contains a set of predefined questions assembled in a predefined order designed to provide answers to questions related to the research; the answers provide data that can be analysed and thus contribute to the research results.

The use of an anonymous questionnaire enables respondents to answer the questions honestly because it reduces pressure to give positive feedback or to show good will towards the research or the researcher. An anonymised questionnaire is likely to achieve a lower response rate than a researcher-administrated questionnaire but, in this case, it was decided that fewer but more honest answers provided by an anonymous questionnaire would be more valuable than a greater number of possibly compromised results (Oates, 2006).

It was hoped that providing the students with a quick and easy self-administrated online questionnaire with no time limit for its completion would boost the response rate, so this was the instrument that was used. The questionnaire had only nine factual questions. Seven were closed questions and offered a range of possible answers. A copy of the questionnaire is shown in Table 9.2 in the

Appendix. One question was half-closed and half-open. Here, the students were asked whether the data structures for which the DSL environment provided help are the most important data structures to have visualisation or if there are other, more important, data structures that ought to be visualised. The last question had an open format in which the students were asked to write their overall feedback about the tool and whether or not it had helped them in learning the concepts of data structures.

For the closed questions, six alternative answers were provided, finely tuned in degrees of agreement. The choice of this format instead of Likert's scale was designed to avoid students choosing the "do not know" answer without really thinking about the question. It forced them to answer more definitely, although this might have alienated those who actually did not know (Oates, 2006). This approach was used because it was important to have definite answers based on qualitative data to support the evidence collected statistically.

The questionnaire was designed to help in answering the following research question.

Q3: Do CS students perceive benefits from aural instructions along with visualisation when studying data structures?

> Students' answers to the questionnaire can provide direct evidence about their perception of any benefits of using aural instructions within the DSL environment. Students were invited to evaluate the extent to which their use of the DSL environment provided a positive effect on their perception of data structure concepts.

5.2.3 Students' Marks

The results of students' assessments in their end of year exam were collected and compared to the data of their usage of the DSL tool to look for possible correlations between the two. Although this experiment did not use pre and post-tests to compare students' progress during the course with their DSL participation, it was hoped that the students' exam results would give an indication of the value of the DSL approach to novice students with lower marks, and how students with different levels of marks had interacted with the DSL tool.

The investigation of the students' assessment results will help in answering the following research question.

Q9: What is the level of achievement of CS students who choose to use the DSL tool the most?

> A comparison of the students' assessment results and the amount of time they spent in engaging with the DSL environment will help in identifying the students who most needed the tool, and interacted with it effectively to actually learn. Obviously, this does not attempt to relate the effect of their use of the environment on their attainment, as there are many other factors that can affect a student's attainment, and that is outside of the scope of this research.

5.2.4 Audio Usage

As this research focuses on the effectiveness of using interactive audio instructions within a visualised learning environment, it was important to capture as much data as possible about the usage of audio in the DSL environment. First, a prototype tool was designed to record if the user chose audio at the log in screen. This would run the tool with aural instructions turned on and disable the use of textual instructions. The second type of data recorded was how many times the student clicked on the "Repeat" button, as this would give an idea about the clarity of the audio. It was expected that the collected data about audio usage would provide insights about the level of audio support and what type of tasks attracted students to use this function.

The data about the audio usage will help in answering the following research question.

Q5: Do students prefer the DSL environment based on visualisation only regardless the type of instruction?

> It is important in this research to identify the impact of aural instructions on students' learning experience. The information about the use of aural instruction should reveal how this type of instruction can benefit those students who use it and if they found it usable and useful. It will also show when students choose to use aural instruction, and when they prefer to use the textual instructions instead.

5.3 Qualitative Data Collection

One of the advantages of the eLearning environment is its ability to trace user navigation and behaviour. Analysing these can help to discover issues about both the tool itself and the students' behaviour while using it (Mor Pera et. al., 2007).

In this research, it was hoped that the qualitative data collected would enable an understanding of the benefits of the DSL environment which integrates aural instructions into a visualised learning setting, how students use the DSL tool, and whether it helps them to achieve what they were trying to do. Three main types of usage data were scheduled: each student's usage of each visualisation task, written feedback within the tasks, and interviews with users. All of the qualitative data collected in this research were acquired during the second study, as the first mini study aimed to collect only quantitative data in the form of response time to instructions.

5.3.1 Reviewing Student Usage

The first step in reviewing the collected qualitative data is to understand what the students generally have done by recreating every session each of them had undertaken. As explained in Section 5.2.1, the design of the DSL environment allowed the automated recording of each student's action in each task, and that provided the data required. The data collected gave detailed information about when the student logged in and for how long. It also detailed each student's inputs and what functions the student used to manipulate the visual components of the task. Later, a snapshot could be taken by using the prototype tool to recreate the image of the visualisation created by the student. It was intended to use the collected images to compare the student's work with their feedback about the task.

5.3.2 Task feedback and Rating

The task usage data discussed in the previous sub-section was designed to be used to identify how the student engaged with the DSL approach, and what content was created. This sub-section discusses a complementary method of collecting qualitative data that provides a better interpretation of what actually happened during the task.

The task interface in the DSL prototype tool offered students the opportunity to provide feedback about the current task. A task rating slider and a free-text feedback area were provided to allow students say what they thought about the task and to communicate any issues they wanted to raise while performing the task. Although the rating could be classified as quantitative data, it was intended to use it to assist the analysis of qualitative data like task usage. Analysing together the textual feedback, the task rating and the usage details would provide more potential to understand and trace each student's behaviour. The presentation of the rating slider however, only included the rating from 1 to 5, and the tool recorded the rating value as 0 by default.

The data from task feedback was designed to help in answering the following research question.

Q8: Do CS students perceive benefits from using the DSL tool to build a mental model of data structures?

> Task feedback can provide information about any benefits that students perceived they gained in their learning within the DSL environment. Students' instant feedback about each task can provide detailed information about their experience with the DSL approach, and the extent

to which they achieved what they expected to accomplish by engaging with the DSL tool.

5.3.3 Interviews

An interview can be defined as a special kind of conversation between two people (Oates, 2006) which allows the interviewer to get information from the interviewee, usually based on a set of planned questions. By asking open ended questions, the researcher can obtain unique answers and more complex feedback that other research methods cannot capture (Oates, 2006). In this study, the interviews aimed to obtain detailed information about the students' involvement in the conducted experiment. The interviews were planned to be conducted at the end of the academic year so that the data they revealed could be used in conjunction with the data collected from the questionnaires to give a fuller picture of the students' experience with the DSL environment.

The conducted interviews were designed to be a combination of artefact-based and semi-structured questions. In the artefact-based part of each interview, the students' will be given their own profile created from their usage data and containing some snapshots of their work, the timings and their written feedback. These profiles will be sent to the students prior to the interviews as a reminder of what they have done, and to help them to remember the tasks and the issues they faced during their work.

The semi-structured questions will cover a list of themes, and follow-up questions will be asked if necessary in order to get in-depth evidence about students' experiences. The themes will cover subjects related to the use of audio and visualisation, and the contributions of the DSL approach towards learning data structures subjects.

In the first part of each interview, the students' will be given the opportunity to comment on the usage data that had been sent to them and to explain what they were trying to do in the shown tasks. Then, they will be asked to say whether or not the tool helped them to achieve their learning objectives for the chosen task.

The choice of participants will be based on varied usage of the tool (limited usage, average usage, and intensive usage) and on their assessments marks, as it is important to see students with a variety of achievement levels. However, due to the limited time likely to be available after the exams at the end of academic year it was known that some participants would not be available for the interviews as many students go home soon after the end of their exams.

5.4 Chapter Summary

This chapter presented the design steps of the research and the experimental methods used to evaluate the effectiveness of the DSL environment. It was expected that all the methods discussed in this chapter would shed light on the use of the DSL tool in the practical sessions of the PDS module, and would gather the information necessary to answer the research questions listed in Table 1.1.

Chapter 6 : Results and Evaluation of the DSL Environment

6.1 Introduction

Having established, through the pilot study discussed in chapter 4, the effectiveness of aural instruction for computer science students in a visual learning environment, this study moved on to investigate the effectiveness of the main feature of the DSL environment, which is visualising the concepts of data structures in conjunction with aural instructions.

This chapter discusses the results of a longitudinal study of the DSL environment, the design of the study and the methodologies used. It also details the data collection methods used, and provides an overview of the students' experience of engaging with the DSL environment during the course of the study, and maps the results to the research questions listed in Table 1.1. Finally, it analyses the results of the study to assess the effectiveness of the approach and the extent to which the results support the research hypotheses.

6.2 Participants

This part of the research was conducted over the second and third terms of the academic year, that is, from 15 January to 25 June 2010. The students had access to the prototype tool whenever they wished to use it. The total number of participants decreased from 30, who took part in the first experiment (see chapter 4), to 27 who actively used the tool in this experiment. The term “actively” means that they used the tool more than once. After reviewing the students’ usage data at the end of the experiment, it was seen that a small group of students had used the tool only once. This indicated a lack of interaction with the tool and these students were excluded from the results and their analysis.

The average age of the students who participated in the experiment was 19.08 years, with only two students aged over 21 years. 19 students had no experience of programming before the course started, five had 1-3 years of experience, and only 3 had more than 3 years of programming experience. Of the 5 females taking the module, 4 actively engaged with the DSL environment, together with 23 of the 39 males.

Students were encouraged to participate in the experiment by emailing them with details about the DSL approach, how to use the prototype tool and how the DSL environment could help them in their study. They were also encouraged to use it by the lab demonstrators, who reminded students of the option of having the tool to hand.

6.3 Experimental setting and tools

As the experiment ran over two academic terms, the DSL tool was made available through a network connection to a CS Department local server (SMART), which is run by the Technology Enhanced Learning (TEL) group. The Information Technology Service (ITS) at Durham University limited access to the DSL tool so that only students using the network computers in the PDS labs had access to it. This provided all the students taking the PDS module unlimited access to the tool.

The decision to use the Durham University CS labs as the experimental environment for the DSL tool was based upon four reasons:

- Students were more likely to access the tool during their practical sessions to help them understand and seek a solution to an assigned practical task.
- The licence for the text to speech engine limited its use to 40 computers at any time.
- The DSL tool is a Visual Basic application and, since the installation process is complicated, the CS labs provided the most suitable environment for this.
- The university's MySQL database service was available to record students' personal information and data usage through a secure connection.

To engage with the DSL environment, students were provided with instructions on how to start the DSL tool. Then, they needed access to a shared folder on the server to run the system executable file. All the lab demonstrators were also made aware of how to run the tool in case they needed to deal with any problems or queries from students.

6.4 Quantitative data analysis (The DSL usage data)

This section will analyse and evaluate the overall approaches to data collection. Both quantitative and qualitative data collection methods were used. The research observed student behaviour by automatically gathering information about their usage. This helped to provide general feedback from students about their overall learning experience. In addition, at the end of every data structures task, they were asked to provide comments on the tool's usefulness. At the end of the experiment, the researcher also conducted a questionnaire with students, as described in Section 5.2.2, to check the conclusions based on the collected usage data and to gain a deeper understanding of their views and experiences in engaging with the DSL approach.

6.4.1 Analysis of automated usage data

As explained in chapter 3, the DSL tool automatically collects detailed information about each student's usage. This data was used to help to answer some of the research questions listed in Table 1.1.

The first type of data collected was information about the usage time and the number of times each student used the system. The experiment recorded the time and date of each student's session. It also recorded which data structures visualisation was accessed and whether any audio assistance was used to accompany the visualisation. Once a student started a visualisation task, the DSL tool started to record the time spent on each task, and all the student's actions.

Figure 6.1 summarises the overall usage data. The mean average time spent on engaging with the DSL tool was about 40 minutes, which equates to one third of a practical session. The results show that, during ten practical sessions over two terms, students spent an average of 3.3% of practical session time using the DSL tool. However, usage by different students varied greatly. The longest time a student spent using the tool was 127.92 minutes (about 2 hours and 8 minutes) and the least time was about 4 minutes. This implies that the results can provided an answer to the research question Q6 "Do CS students choose to use the DSL tool while studying data structures?"

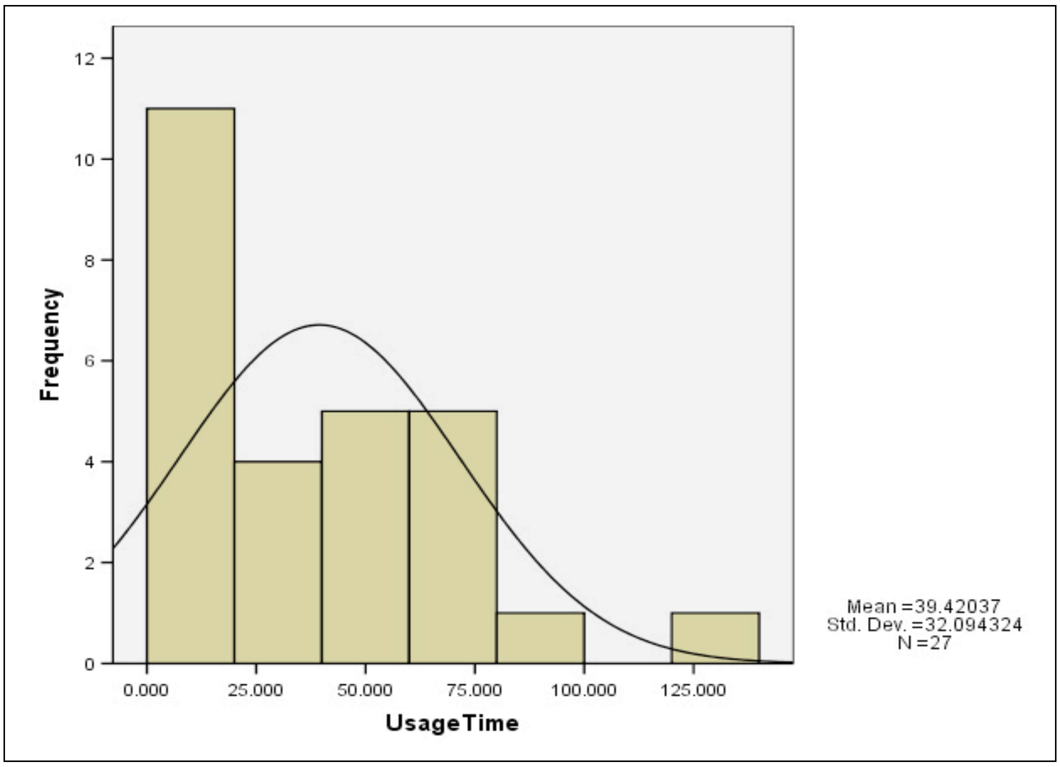


Figure 6.1: Students' usage time

With regards to the number of tasks undertaken by students, Table 6.1 shows that students used the DSL tool with all the visualisation tasks. They logged in to the tool approximately 4-6 times and used the visualisations tasks more than once. This suggests that the students were interested in the DSL environment.

Although the previous data provides an overview of the tool's usage, more investigation was needed to assess what sort of visualisations were more likely to be used by students and what was their effect. To answer this, the DSL environment was designed so that the prototype tool records each student action while working on the task.

Task Type	COUNT
Linked Lists	50
Binary Search Trees	55
Binary Tree Traversal	66
Total Usage	**171**

Table 6.1: Task Type Count

Table 6.1 shows that the students engaged with the Binary Tree Traversal (BTT) visualisation task the most and with the Linked Lists (LL) visualisation the least, but the DSL tool was well used in all three tasks.

Table 6.2 provides further details of the collected data and shows the number of actions performed on each visualisation during the course of the task. In order to avoid discrepancies in the system's usage data, all the tasks where less than 5 actions were performed were eliminated from the results. This was because, after checking these instances, it became clear that they were not genuine interactions with the tool and the students did not perform valuable actions with it. Thus, it was unrepresentative data. Table 6.2 shows that the BTT task had the most student interactions with the tool. The results enable an answer to be given to the research question Q7 "Of the three data structure types used in this study, which do CS students select to explore through the DSL tool?" The answer is that the students made substantial use of the DSL tool to help them study all the three types of data structures. They ran a total number of 142 tasks and, within these, they performed nearly 1,500 actions while interacting with the tool. However, the BTT visualisations attracted the largest number of both runs and actions and these accounted for 38% and 48% of their respective totals.

Task Type	Task Total Runs	Total number of actions
Binary Search Trees	46	367
Linked Lists	42	404
Binary Trees Traversal	54	710
Total Effective Usage	**142**	**1481**

Table 6.2: Effective usage of DSL approach

6.4.2 Task Feedback

As explained in chapter 3, the DSL environment also allowed students to rate each task they engaged with and to produce feedback that was specific to the task. For this, a rating scale was offered with each visualisation task. The scale allowed the student to rate each task with 1 to 5 stars, with 1 being poor and 5 being excellent. Out of the total of 142 effective tasks undertaken, the students used the rating scale on 60 occasions (23 BTT task, 19 BST task, and 18 LL task). The overall rating for the three visualisation tasks is shown in Figure 6.3. The Binary Search Tree (BTT) task achieved a higher rating than the other two tasks, which correlates with the previous data about the greater interaction with that task.

There was no instance of a feedback rating of 1 (poor), and most of the students claimed to have found the approach useful. However, 21 of the 60 feedback scores were 3 (neutral).

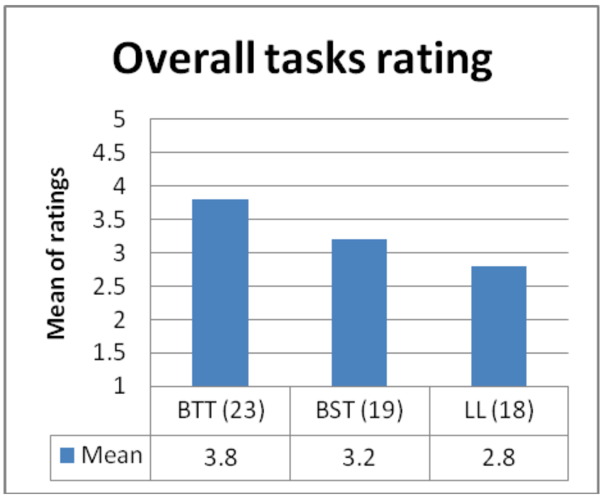


Figure 6.3: Students' rating of the DSL tool

In addition to the rating, written feedback could be provided and, in general, this showed appreciation of the contribution of the DSL approach to an active learning environment. This will be discussed in detail in Section 6.5. Both types of feedback showed that students had few technical issues with the prototype tool and the researcher took immediate action to deal with them. The DSL tool also recorded snapshots of students' work when they chose to save images. The analysis of those images will be also discussed in Section 6.5.

6.4.3 Audio Usage

The DSL environment permitted the prototype tool to gather some additional and important information about how students interacted with the tool's audio functionality. As will be seen below, this extra information helps to answer the research question about the effectiveness of using audio with visualisation.

Chapter 5 of this study discussed the use of audio in this research and how it would be measured. Students were given the choice to use the tool with or without aural instructions. At the login screen, students could choose either to have aural or written instructions, but they could not have both. However, there is a known risk of a student choosing to use aural instructions without having headphones on, or taking the headphone off when switching between visualisation tasks. The only way used to avoid the risk, was by continuous monitoring of students usage of the DSL tool during the practical session by the main researcher conducting this research. However, there was no case found where students have reported that they are using audio without actually using it.

Table 6.3 shows the percentage of audio usage with visualisation versus using visualisation alone. Again, these results were based on the number of students who engaged actively with the DSL tool, as defined in Section 6.2. The overall usage of audio in all the tasks performed by students was 54.11%. However, this data needs further investigation before any conclusions can be made.

Visualisation Type	Percentage of Audio Use
Binary Search Trees	56.52%
Linked Lists	59.52%
Binary Trees Traversal	46.30%
The overall usage of audio	54.11%

Table 6.3: Percentage of Aural instruction usage against Textual instructions

Looking back at the data generated by the students' engagement with the DSL approach, it was noted that most of the audio usage (77.77%) was from tasks that generated less than ten actions. This means that most students used audio in short tasks, whilst students involved in longer tasks requiring a high number of actions were less likely to use audio with a visualisation. Again, further investigation into these results will be discussed in the qualitative analysis in Section 6.5. An overview of the automated usage results indicates that there is a positive relationship between the extent of students' use of the DSL tool in a task (and giving it a higher rating) and the use of aural instruction.

6.4.4 Analysis of questionnaire data

An online questionnaire about their learning experience with the DSL environment was sent to the participant students at the end of the experiment. Only 16 of the 30 students responded. This low percentage (53%) may have been due to the timing being near the end of term. Although the survey was open to responses for a 3-week period, many students left the University as soon as they had finished their exams. The summary of the questionnaire results is shown in Appendix C.

The results of the returned questionnaires were as follows:

- As shown in Table 6.4, the majority of students, 93%, gave a positive response. 37% of these “strongly agreed” that the DSL tool was easy to use. 6.3% “slightly disagreed” with this statement.

The learning tool interface was easy to use	Response Percent	Response Count
Strongly Agree	37.50%	6
Agree	43.80%	7
Somewhat Agree	12.50%	2
Somewhat Disagree	6.30%	1
Disagree	0.00%	0
Strongly Disagree	0.00%	0
	Answered questions	**16**
	Skipped questions	**0**

Table 6.4: Students' responces to quiestionner question Q1

- As shown in Table 6.5, the overall feedback about the clarity of the audio instruction was positive, also at 93%. However, 31.3% of students only "somewhat agreed" with this statement.

The audio instructions were clear and easy to follow	Response Percent	Response Count
Strongly Agree	25.00%	4
Agree	37.50%	6
Somewhat Agree	31.30%	5
Somewhat Disagree	6.30%	1
Disagree	0.00%	0
Strongly Disagree	0.00%	0
	Answered questions	**16**
	Skipped questions	**0**

Table 6.5: Students' responses to questioner question Q2

- As shown in Table 6.6, a high percentage, 43.80% of students, "strongly agreed" that the Binary Tree Traversal visualisation was a useful and important task to enhance and test their knowledge of the subject. No negative feedback was given about this part of the study.

The Binary Tree Traversal task was useful and important to test and enhance your knowledge about tree traversal	Response Percent	Response Count
Strongly Agree	43.80%	7
Agree	25.00%	4
Somewhat Agree	31.30%	5
Somewhat Disagree	0.00%	0
Disagree	0.00%	0
Strongly Disagree	0.00%	0
	Answered questions	**16**
	Skipped questions	**0**

Table 6.6: Students' responses to questioner question Q9

- As shown in Table 6.7, 37% of students thought there was a need to visualise other data structures, such as the Self-balancing Binary Search Tree (AVL Trees), Graphs, and Control Flow Graphs. However, the remaining 63% of students thought that the visualisations actually selected for treatment by DSL approach were the most important data structures.

The provided visualisations coverd the main data structures that considered important in this module	**Response Percent**	**Response Count**
Strongly Agree	12.50%	2
Agree	50.00%	8
Somewhat Agree	31.00%	5
Somewhat Disagree	0.00%	0
Disagree	6.30%	1
Strongly Disagree	0.00%	0
	Other Data Structures	4
	Answered questions	**16**
	Skipped questions	**0**

If you think there are more important data structures to visualise, please list the in the empty text box

1. AVL Trees was more important
2. Control flow graph
3. AVL Trees
4. More specific types of tree (AVL trees, etc)

Table 6.7: Students' responses to questioner question Q4

- As shown in Table 6.8, 34% agreed or strongly agreed that the visualisations helped them to create a mental model of these data structures.

The visualisations helped in creating mental model of the proposed data structures	Response Percent	Response Count
Strongly Agree	37.50%	6
Agree	31.30%	5
Somewhat Agree	25.00%	4
Somewhat Disagree	6.30%	1
Disagree	0.00%	0
Strongly Disagree	0.00%	0
	Answered questions	**16**
	Skipped questions	**0**

Table 6.8: Students' responses to questioner question Q5

The questionnaire results provide evidence that the students perceived benefits from using aural instructions and visualisation by engaging with the DSL tool. 93.8% of students reported that their use of the DSL tool helped them to create a mental model of the three data structures.

In the written feedback about the DSL experience, one student reported that the DSL tool "was a very easy to use tool that provided a good visual aid for linked lists and binary search trees." This comment shows a deep understanding of the approach and his positive interaction with it. However, he also added that "the linked list program was quite buggy and sometimes left the program prone to crashing." It should be noted here that there are no records of how many system crashes occurred because the data was saved only upon normal and error free closure of a visualisation task.

There were mixed reviews about the usage of audio to support visualisations. Some found it good and useful, for example, "audio instructions and descriptions were very useful, as well as the ability to add and remove nodes into an existing list." The students who experienced technical issues with integrating audio and visualisations tended to be deterred from using it, with one student explaining, “Audio didn't work first time, and I haven’t used it since then.”

The questionnaire was also an opportunity for the students to share their opinions on how the DSL environment could be improved. Suggestions included adding extra features to the prototype tool, for instance, “I think the user interface could be made easier to use and more intuitive”, or better ways of engagement, such as “it would be helpful if it was integrated into lectures / practical’s.” The latter comment suggests the need for more integration between external learning resources, like the DSL environment, and the PDS module’s contents.

6.4.5 Students Marks vs. engagement with the DSL environment

As explained in Section 5.2.3, the student participants in this experiment did not do a pre-test and post-test to see if the DSL environment had a direct effect on their learning. This was because there were other types of learning inputs to the PDS module so it would not have been possible to isolate the effect of the DSL environment. In any case, the DSL environment was not part of the PDS module and it was not considered a formal learning approach. Instead, the research looked at students’ results in their PDS end-of-year assignment to investigate possible correlations between these and their engagement with the DSL environment, keeping in mind that correlation does not necessary mean causation, and the researcher had no information about the students’ abilities and weaknesses.

To compare students’ assessment marks with the duration of their engagement with the DSL tool, Figure 6.4 shows the linear regression of the scattered marks against engagement with the DSL approach. This shows, in general, that students who spent more time using the DSL tool obtained lower marks in the end-of-year assessment. This result may suggest that less able students used the DSL tool as an additional resource to help them to understand the concepts of data structures, rather than that their use of the tool contributed to their lower marks. If true, this finding confirms the need for learning methods like the DSL to provide students with an extra support system if and when they need it. Although there is no clear indication about whether or not using the DSL tool affected the students' marks, it is important to note that the less able students used the tool repeatedly, and found it valuable to their study. It is also possible that these students could have gained even lower marks if they had not used the DSL tool. However, the lecturer who taught the data structure module feedback about the DSL environment clearly indicated that students have benefited from their engagement with this environment. The lecturer also expressed his interest in using the DSL environment for the next coming year.

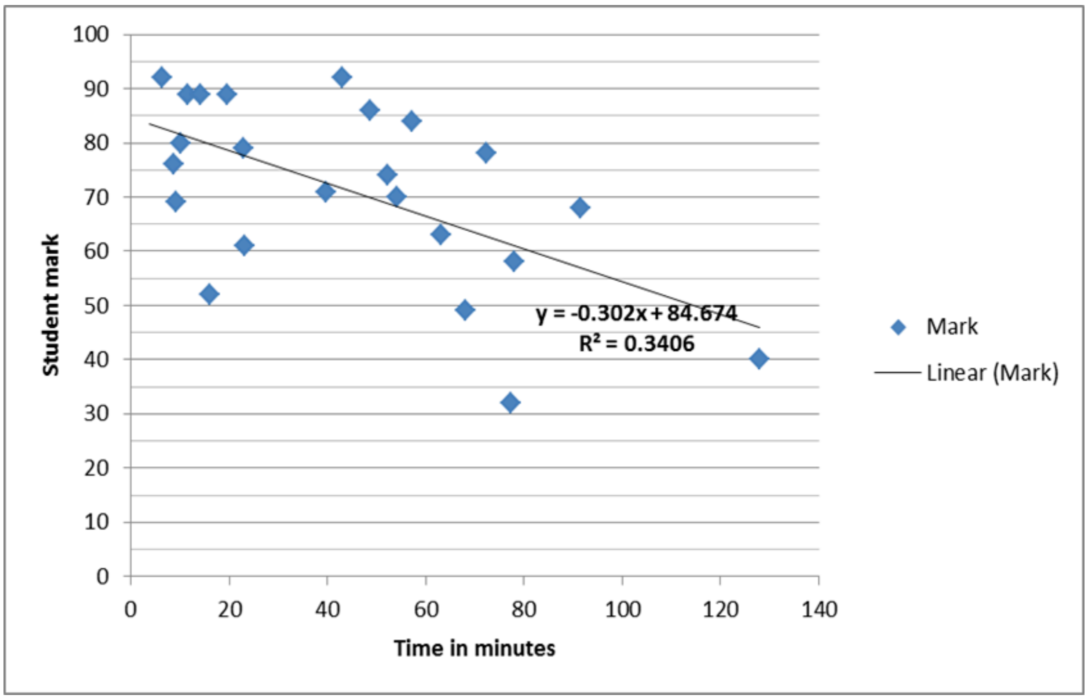


Figure 6.4: relation between student mark and time spent using DSL tool

6.4.6 Summery

Section 6.4 has presented and analysed a variety of quantitative data generated by the DSL environment and the student questionnaires. The next section will discuss in detail a range of qualitative data and look at some examples that can help in understanding the results of the quantitative data. Moreover, further discussion on the data is reserved for Chapter 7.

6.5 Qualitative data analysis

Large amounts of quantitative data were generated from automated usage, task feedback, recorded audio usage and questionnaires and these were discussed in Section 6.4. However, answering the research questions listed in Table 1.1 largely depends upon analysis of the qualitative data. This section will discuss the qualitative data collected in this study. The experimental approach used in this research differed from other common experimental methods in that a longitudinal method was used. This was because the experiment stretched over two terms and the research was trying to assess educational value, not short-term effects upon students' responses that may be quickly forgotten.

The qualitative data collected in this study comprised of the students' feedback on each task they performed, a collection of snapshots of their work, written feedback and interviews with participants.

The qualitative data collected and analysed achieved two main objectives:

- It supports the results obtained by analysing the quantitative data discussed in Section 6.4

- It provides data that allows triangulation to be used, that is, using the results obtained from one method to support or question the results of the other method. Triangulation is likely to provide better evidence with which to answer the research questions in Table 1.1

6.5.1 Analysis of individual task comments

Section 6.4.2 of this research looked at students' rating of each task they performed. For each task, students rated the benefit of each task they undertook using a five point scale and they were also asked to write a short comment about it. The collected comments were filtered so that repeated comments were ignored (total of 9 comments). Short comments without any useful feedback (e.g., "nice tool", "good work!" and "Note") were excluded, leaving a total number of 55 different and useful comments. Appendix B and C shows the complete list of students' comments. The comments were then divided into two categories, those on non-technical issues and those of a technical nature. The technical issues raised by students were dealt with immediately and errors were fixed straight away.

6.5.1.1 Non-technical feedback

Most of the non-technical feedback reflected the students' general ideas about the use of aural instructions with visualisation in an interactive learning environment, or they referred to specific features. For example, a student stated "Audio is good - it helps greatly with the task …" Another said, "this part has provided me with a deep knowledge about the linked list as I was finding it hard to acquire information from text books … view as array and wiki functions are great ideas!" Such comments showed a good level of engagement by students with the DSL approach and reflected their understanding of the approach's purpose as well as highlighting its strengths.

The other type of non-technical feedback was students' reflections upon their own learning experience when interacting with the DSL environment. As an example, a student wrote about his use of the prototype tool, "Excellent tool, especially for beginners…very interactive ..." Also there were comments like, "Good enough for first time use" and "audio instructions and descriptions are very useful, as is its ability to add a node into an existing list."

Although most of the non-technical feedback was positive, some students raised issues that need noting. A student commented on the absence of a guide on how to use the tool by saying "… this section demonstrated tree traversal well. However, a video tutorial on how to use it would have been the "icing on the cake." Another student suggested that the DSL approach should have included more important data structures, saying "… useful for visualizing the objects, though possibly a function for looking at classes and more difficult to grasp topics such as inheritance……. would be more useful than understanding simple objects." The DSL tool did lack a full guide, but it did actually address visualising "inheritance" by introducing a simple inheritance in the LAO task, as described in Chapter 3 and illustrated in Figure 6.5.

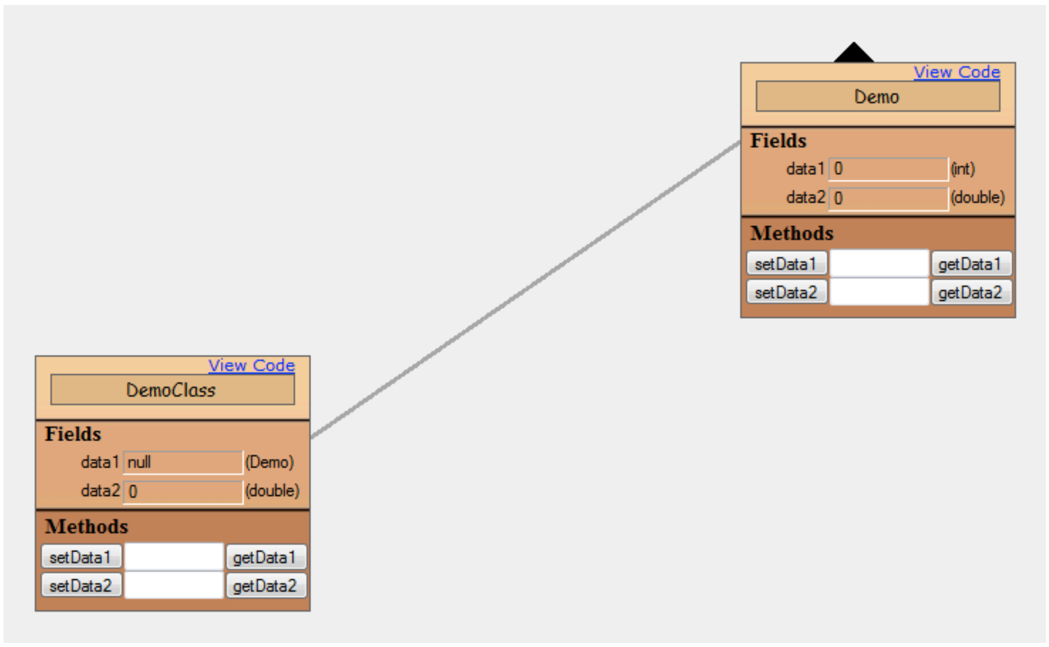
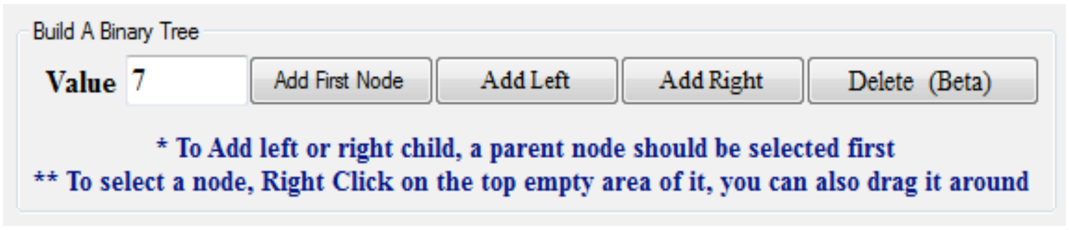


Figure 6.5: Showing simple inheritance example

The non-technical feedback provided information about the benefits that students perceived that they gained from this approach to learning. Task feedback, in particular, provided detailed information about their experience with the DSL environment. It was clear that students had engaged well with the DSL approach, and had understood the DSL concept. This feedback also supported the preliminary results and provided an impressive rating for the tool's qualities.

6.5.1.2 Technical Feedback

In a few cases (27), students reported technical problems experienced while they were working on a task. Again, Appendix C shows a complete list of such feedback. The technical type of feedback was monitored closely and, in many cases (19), a resolution was produced immediately to fix the problem reported by a student. For example, as shown in Figure 6.6, the onscreen notes were clarified immediately after receiving feedback which said, "… maybe make it clearer that you can double click elements to display more data."

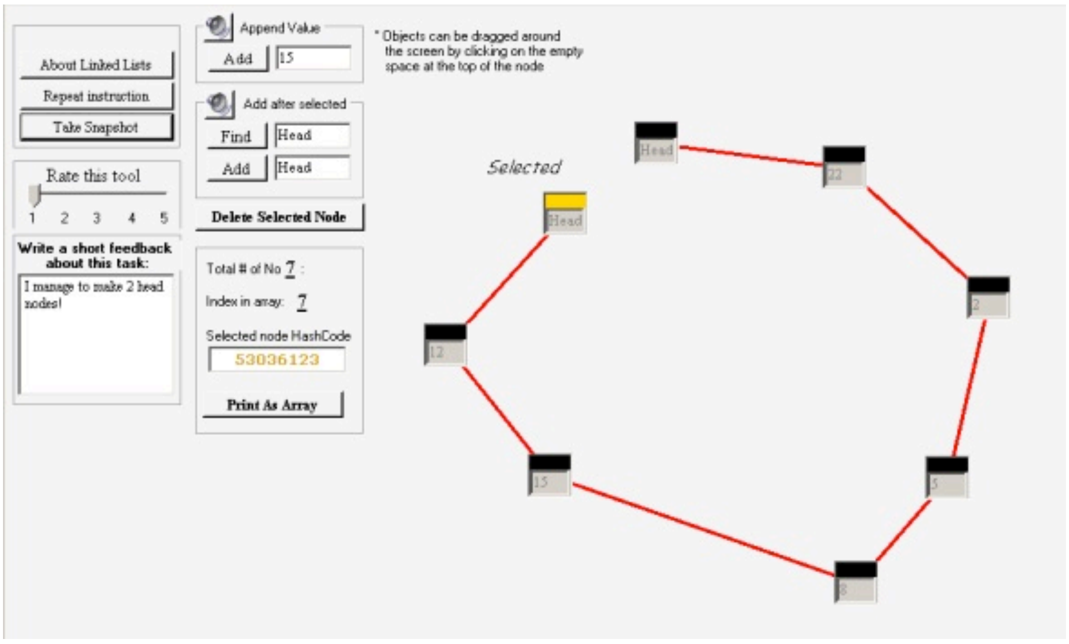


* To Add left or right child, a parent node should be selected first
** To select a node, Right Click on the top empty area of it, you can also drag it around

Figure 6.6: Enhancing onscreen help after suggestion by students

Although a large part of the technical feedback related to the prototype tool's functionality, most of which was resolved without delay, some comments raised important issues about the learning content visualised in the DSL tool. For example, it was pointed out that the visualisation used did not give enough information to aid greater learning about Binary Trees. One student said, "… the way in which nodes are added to the tree is not clear to me and so the tool doesn't seem that useful until you can construct a tree, exactly as you want it." Another claimed, "I would like to be able to edit/delete objects, for example being able to change the object name."

In addition to recommendations for enhancements to functionality, some feedback reported the lack of specific functionality to help them in learning the concepts of the offered data structures. A user wrote, "… might be better if somehow the previous task could somehow import the binary trees created to use in this exercise …", showing that the student was keen to make the tool even better by linking two tasks together. Similarly, a student wanting better quality visualisation wrote, "… an animated graphical representation of adding and removing elements when it happens would be useful…” In another example from the Linked List task, a student reported that “… I think it would have been really good to actually know which of them are tails because some people still get confused between heads and tails and think tails is ONLY last value in the list …”, while another expressed his misunderstanding of how the DSL tool works by writing, “I managed to make 2 head nodes!” In the latter example, the student tested the tool’s validation by adding a node with the text value “Head”, and thus he discovered a technical fault that can occur in the Linked List visualisation task. A snapshot of what this user did is shown if Figure 6.7.



Append Value: Add 15

Add after selected: Find Head; Add Head

Total # of No 7

Index in array: 7

Selected node HashCode 53036123

Write a short feedback about this task: I manage to make 2 head nodes!

* Objects can be dragged around the screen by clicking on the empty space at the top of the node

Figure 6.7: Two-Head list created by a student

6.5.2 Analysis of Interviews

This part of the research explores the results of interviewing three student participants together in a group. Although this is a small sample of the participants, the interviewees provided valuable feedback about student engagement with the DSL approach. However, because of the small sample size, generalisations cannot be made about the student population at large.

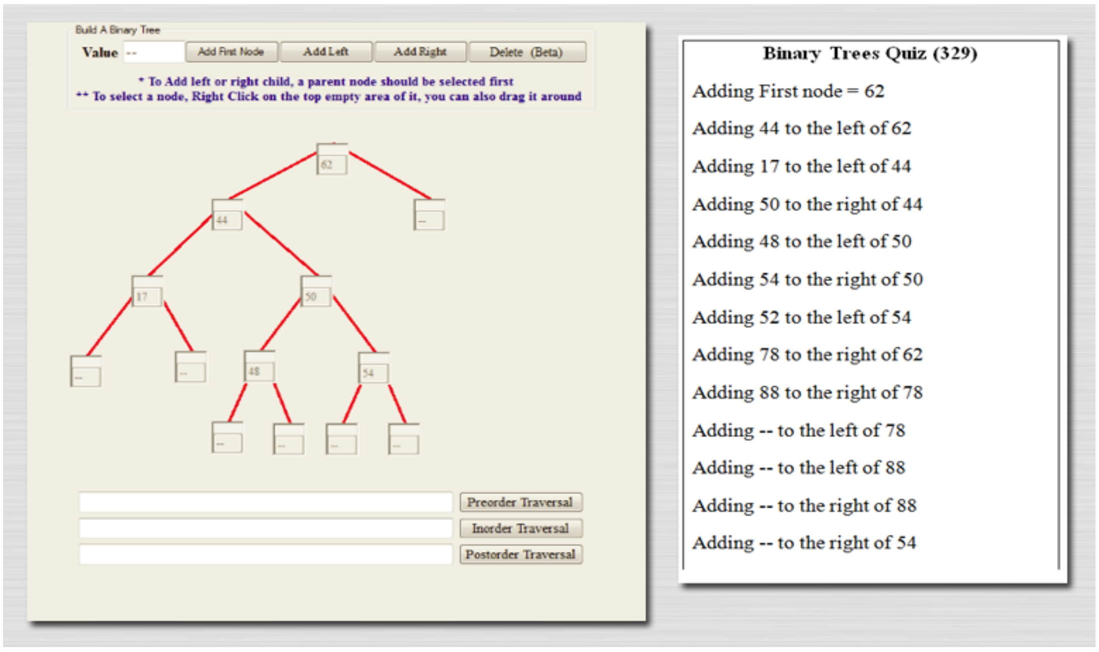
The students were carefully selected from among those who volunteered to be interviewed about their experience of using the DSL environment and after reviewing all the students' usage data. The selection criterion was discussed in Section 5.3.3. There was a wide variation in the students' assessment marks, experiences and feedback, as shown in Table 6.9.

	Participant 1 (P1)	Participant 2 (P2)	Participant 3 (P3)
Age	19	18	18
Gender	Male	Female	Male
Programming Experience	Less than a year	2	3
Time Spent on DSL	78 Minutes	57 Minutes	92 Minutes
Practical Mark	58	84	68
Task Performed	19	10	13

Table 6.9: Interviewees' quantitative data

Before the interview, a profile of each student's usage was sent out to them to remind them of what they had done. A sample is shown in Figure 6.8 below. Each profile also contained snapshots of the DSL tool screens that the student had generated or they were replicas based on the student's action list. In addition, the profile contained all the written feedback the student had generated throughout the experiment.

At the beginning of the interview, each student was asked if the results of his/her profile analysis correlated with what they actually did when they performed the given task, and his/her feedback was explored. This method of using artefact based interviews was discussed in Section 5.3.3. Figure 6.8 shows a sample of P1's usage profile.



* To Add left or right child, a parent node should be selected first
** To select a node, Right Click on the top empty area of it, you can also drag it around

Binary Trees Quiz (329)

Adding First node = 62
Adding 44 to the left of 62
Adding 17 to the left of 44
Adding 50 to the right of 44
Adding 48 to the left of 50
Adding 54 to the right of 50
Adding 52 to the left of 54
Adding 78 to the right of 62
Adding 88 to the right of 78
Adding -- to the left of 78
Adding -- to the left of 88
Adding -- to the right of 88
Adding -- to the right of 54

Figure 6.8: Sample of P1 profile data

Then the following six questions were put to them. Depending on the nature of the response, the researcher asked for elaboration in order to gain as much insight as possible.

1) What did you aim to achieve in this task?

Each of the interviewees reported that they had completed the assigned task. Two of them (P1, P2) said that they were trying to evaluate how the tool visualises data structures using an example from the lecture notes. P3 described the task as something he needed to do as part of his in-lab assignment. For example, with reference to the tree traversal task shown in Figure 6.8 and talking about adding the "--" values to the tree, he said, "I was doing practical work at the time. I wanted to visualise a tree in the programming assignment… I was trying to work out whether there are empty children or not."

The answers to this question reflect the students' intentions when first engaging with the DSL environment. It is likely that many students first made use of the learning tool in order to test its value to them before deciding whether or not to use it again.

2) Has the approach helped you in achieving what you were trying to do when working on a task?

P1 stated that it helped him in the visualisation of data structures, and that was the reason he used it. P2 agreed but claimed it lacked some functionality, saying, "if you can export the images of the created objects to use outside the tool, that will be handy." Actually, this feature is available, as discussed previously. P3 expressed satisfaction with the DSL approach in more detail, saying, "… it was quite handy actually! Being able to see graphical representation of what you were looking at, seeing how to add or remove things from entire structure. I thought that quite good actually." These comments reflect an overall satisfaction with the DSL approach among the interviewees, and a belief that it had helped them in achieving the purpose of the task. These comments were supported by reviewing the students' usage profiles that showed that they engaged with the tool, created a set of visualisations and manipulated them in a logical manner. Thus, it seems that the DSL tool offered a valuable solution to problems that students encountered.

3) With regards to the traversal task, why did you use it more to build trees rather than checking your knowledge by taking the quiz?

None of the three interviewees had taken the quiz to test their knowledge about traversal, although other students had used it and some had used it frequently. It was important to investigate whether this functionality was not relevant to their needs or whether they had simply missed it. P1 stated, "… it wasn't what I wanted to do at the time. When I used the tool, I used it primarily to construct visualisation. I wanted to build trees more than learning traversal. When I studied traversal, I looked at the node that I had." P2 agreed, saying that, after she had tried it once, "… I couldn't understand the quiz part of the traversal task. I might have been entering the data in a wrong format I think …" P3 said, "I did a few traversal exercises. I tried the quiz but it wasn't clear what I needed to do. It might've been something to do with my input; I misplaced the data a few times. I don't think it was an error with the tool." Although there was a range of reasons for not taking the quiz, the most important point here is that both P2 and P3 showed were confused about how the task worked. While they blamed themselves for this misunderstanding, it is the responsibility of the system designer to ensure that students can understand the objective of each task and the instructions for performing it. This is an example of the limitations of the environment, and future work should include running usability checks before students start interacting with the environment.

4) In respect to the use of interactive aural instructions, what did you think of this functionality?

Answers to this interview question were crucial to answering Q3 of the research questions “Do CS students perceive benefits from aural instructions along with visualisation when studying data structures?” and whether the use of audio with visualisation technologies can facilitate student learning.

All interviewees agreed that the instructions were clear and easy to follow. Reflecting the concept of learning styles, P1expressed a preference for reading instructions; indeed all of them explicitly said, “… it is a matter of personal preference.” P3 focused upon the visualisation itself because he found that the audio instruction slowed him down.

Interestingly, P2 linked the use of audio to an experiment during the first term while the students were studying Binary Numbers. That was an experiment in another researcher’s study, which is not related to this research. However, P2’s comment was interesting and useful as she explained the problems, saying, “… well, if you look at instructions like in Janet's class, instructions had to be used to help in constructing binary numbers. We didn't know how to do it and we had to follow instructions. The instructions were too fast, and we had to go back and forward to listen again. At that time, it would've been better if it was combined with words rather than audio.” The student followed with “… just quick short instructions may have helped. But not all the way audio.” Based on these students’ answers, the benefits of the use of audio can depends on various factors, such as a student’s own learning style, external factors or previous experience with audio based materials, the usability and accessibility of the learning tool, or simply the availability of headphone speakers for the students at the time of the practical session.

5) Do you think the DSL environment helped you to better understand the concepts of data structures?

Generally, the students were positive about the value of the environment, saying "… helped in reinforcing concepts" (P1), and "Yes, particularly the traversal exercise…" (P2). P3 believed the DSL approach helped him in his learning and added "… it showed any important information by either displaying it on screen, spoken or even in a text box; it made it easy to interact with." The answers showed that, overall, the DSL environment was useful in learning the concepts of data structures, and had a good effect on their learning experience.

6) Do you think an environment like this should be used as part of the module to help you in what you are studying?

This question was added to the question list after the module leader expressed the intention to use the prototype tool again in the following academic year. The module leader thought it would be useful for the students to have interactive visualisation with aural and textual support, but it was also important to know what the students thought about it. The three interviewees agreed with the module leader's intention, and each highlighted a benefit of the approach. For this question in particular, it is important to present each interviewee's response as it showed a deep understanding of the purpose of the DSL approach and an interest in enhancing such technology for future use.

P1 stated that, "if a lecturer demonstrated how to fill the tree using a visual tool, or how students could build a tree and had the homework based upon that, that would be very helpful. Students who missed the class will benefit from it too." P2 stated that “I think the main advantage will be to make the understanding of the subject faster than any other methods I have used.” Finally, P3 said, “I think potentially yah, because it gives you a quick way of checking your work, and making sure your trees are correct, and your linked lists are correct. It will certainly speed up teaching these things, because you wouldn’t have to read as much to gain the same level of information. It shows you instantly the changes that are made.” These students clearly recommended that the DSL environment should be used in future years, and that it should be part of the module’s supporting materials.

6.6 Summary

This chapter has discussed the results of the experiment conducted within this research about the use of aural instructions with visualisations in an interactive learning environment. It explained the design of the experiment, the research methods used and the data that were collected. The next chapter will relate these results to the research questions listed in Table 1.1.

:

Chapter 7 :

Discussion of Research Results

7.1 Introduction

This chapter analyses the impact of using aural instructions with visualisation in an interactive learning environment, and the overall results of the students' experience with the DSL environment. It also evaluates the collected quantitative and qualitative data. This analysis and evaluation will be used to answer the research questions proposed in Table 1.1.

The chapter will discuss the research hypotheses and questions in the order in which they were presented in Table 1.1. It will include a discussion of the significance and implications of the research findings, as well as their limitations.

7.2 Reductions of cognitive load and improving student engagement

7.2.1 Introduction

This section discusses the impact of using aural instructions and visualisation on learners' cognitive load. It deals with the first research hypothesis, which is, "Reducing cognitive load improves student engagement and outcomes when learning data structures." It will also gather the results of the study to answer the following research questions:

- Q1. What existing research evidence is there that the simultaneous use of aural instruction along with visualisation reduces cognitive load when learning data structures?
- Q2. Does the combination of aural instruction and visualisation reduce students' response time for task completion compared with textual instruction and visualisation?

7.2.2 Research Method

Two main research methods, a systematic review and a cross-sectional experiment, were used to gather data to answer these two research questions.

In relation to Q1, the systematic review investigated previous research in the field of multimedia learning and cognitive load. The main focus was on studies that proposed a presentation of visual content in synchronization with aural narration to produce an efficient allocation of cognitive memory resources and maximise the amount of information gained by learners. Section 2.6 presented a range of studies that investigated this approach, and specifically response times to short instructions.

The cross-sectional study was used to answer Q2. Chapter 4 presented this mini study that investigated the use of the DSL tool to measure the CS students' response times to aural and textual instructions. It is assumed that shorter response times indicate a reduced cognitive load.

7.2.3 Results and Discussion

The results of the systematic review were previously presented in Section 2.6. They showed that, based on cognitive load theory, the use of visual and aural presentation simultaneously is likely to result in a lower cognitive load on a learner's working memory (Sweller, 1994; Mayer, 2001; Tudoreanu, 2003; Ando, 2008). The review also provided evidence that the use of textual presentations with visualisation may distract learners by splitting their attention between the visual presentation and the text, which can be considered as another form of visual presentation (Sweller, 2002; Kalyuga, 2006).

The literature also investigated the relation between cognitive load and response time to instructions or questions. It showed that, in intelligent tutoring systems, the response time to aural instructions positively correlates to cognitive load and that a longer response time indicates lower performance and less interaction with the material (Beck, 2005, Hensler, 2006). Conversely, longer response times to giving a correct answer to a complex question indicates a higher cognitive load (Dror, 2005; Khawaja et al., 2007). The hypothesis in this research however, is that if both correct response and the response time indicate or imply that cognitive load is reduced, and if correct responses are received from all learners, then response times can indicate whether cognitive load is increased or decreased, that is, that a greater response time indicates an increased cognitive load.

Based on that investigation, the requirements of a prototype tool that could bring together an effective approach for reducing cognitive load were presented in Chapter 3. Additionally, Chapter 4 presented a cross-sectional mini study that was conducted to answer the second research question, "Does the combination of aural instruction and visualisation reduce students' response time for task completion compared with textual instruction and visualisation?" The results of that study were discussed in Section 4.4, and summarised in Figure 4.4. The results showed that, when synchronised with visualisations of the concepts of data structures, learners record a statistically significant higher response time for textual instructions than for aural instructions. Based on this result, the conclusion was drawn that the use of aural instruction with visualisation can reduce students' response time and, based on the results of the systematic review, it was concluded that the reduction of response time indicates a reduction of cognitive load.

Thus, this thesis has demonstrated positive support for the hypothesis H1 that "Reducing cognitive load improves student engagement and outcomes when learning data structures."

7.2.4 Threats

Two threats to the validly of the first research hypothesis investigated in this section were identified. The first threat is the validity of the resources used for the systematic review. No study was found that explicitly states that a reduction of CS students' cognitive load can be accomplished by using short aural instructions with visualisation in a multimedia learning environment that aims to teach the concepts of data structures. In addition to being interesting and potentially useful to teachers and learners, this study's focus on this gap in the research literature fulfils the requirement for uniqueness of the research, which is a necessary element of doctoral research. Based on that, the second threat to the validity of the hypothesis is the adoption of response time as the sole dependent variable for the mini study. A discussion of the procedural confounds were presented in Sections 4.2.3 and 4.2.4.

7.3 Use of aural instructions and students perception of data structure concepts

7.3.1 Introduction

This section investigates the impact of using aural instruction with visualisation on the process of learning the concepts of data structures. It presents the second hypothesis of the thesis, that is “The use of aural instructions in teaching data structures to Computer Science students has a positive effect on student perception of data structure concepts.” The purpose of this investigation is to gather results to enable answers to be given to the following research questions:

- Q3 Do CS students perceive benefits from aural instructions along with visualisation when studying data structures?
- Q4 Is there a relationship between visualisation type and CS students’ choice of instruction type?
- Q5 Do CS students prefer to use the DSL tool based on visualisation types only regardless the type of instruction?

7.3.2 Research Method

To answer the three research questions in this section, a triangulation method is used to make sure that data from one research methodology can be used to explain the results of the other.

Students’ perception of the benefits of aural instructions is measured by their responses to the questionnaire. The questionnaire results, presented in Section 6.4.4, provide a general idea about the students’ perceptions of the DSL approach.

Students were asked about the clarity of the audio instructions in order to judge whether students found any difficulties in understanding the instructions or problems with the voice that was used, as shown in Section 6.4.3.

A qualitative data collection method was employed to obtain further information to answer the research questions. Students provided written feedback about their experience with the DSL tool. This feedback provided information about how students perceived the use of aural instructions in the learning environment. Interviews were conducted with three selected students to obtain a better insight and evaluation of how the approach was received by students. Section 6.5.2 presented an analysis of each student's comments about the use of aural instructions.

7.3.3 Results and Discussion

To answer the research question Q3 "Do CS students perceive benefits from aural instructions along with visualisation when studying data structures?" Section 6.4.4 presented students' responses to the questionnaire at the end of the academic year. The responses showed that students agreed that the DSL approach had helped them to build mental models of data structures and the aural instructions were clear and easy to follow. Moreover, they offered the opinion that the integration of the DSL approach with the lecture materials could make it even more beneficial. Some students hoped that more visualisation tasks could be added to the tool. Thus the answer to research question Q3 is that CS students do perceive benefits from aural instructions along with visualisation when studying data structures.

The research question Q4 asked "Is there a relationship between visualisation type and CS students' choice of instruction type?" Investigation of this question aimed to provide evidence that the use of aural instructions has a positive effect on learning data structure concepts in general, but also that students might tend to use aural instruction more in learning harder-to-understand concepts. Section 6.4.3 presented the analysis of the usage of the aural instruction format. Students who were actively engaged with the DSL tool used aural instructions in 54% of the tasks they performed. However, the data shown in Table 6.4 showed that there were no significant differences in the use of the two different types of instructions in the three different visualisation tasks. Based on that evidence, no conclusion can be drawn about whether the difficulty level of the task affects the students' choice of instruction type.

The research question Q5 asked "Do students prefer the DSL environment based on visualisation only regardless of the format of instruction?" To answer this question, it was necessary to assess the impact of aural instructions on students' learning experience. As shown in Section 6.5.1, students' comments and feedback about their usage of the DSL environment included feedback about issues faced while using the DSL tool, suggestions to enhance it, and reflections upon their learning experience. Students reported that they had benefited from using aural instructions and that they were very useful and interactive.

Moreover, in the student interviews reported in Section 6.5.2, the interviewees agreed that the aural instruction was clear and easy to understand. However, they had some reservation about using aural instructions all the time. One stated a personal preference for textual instructions and the way in which this was expressed revealed that she was conscious of her own preferred learning style. Though investigation of possible relationships between students learning styles and the DSL environment was not part of the study, it is recommended, especially in the light of the discussion in Section 2.2.7, that this should be included in any future work.

The results showed that aural support was used on almost half of the occasions when a student used the DSL tool. This result was investigated further by constructing a timeline of students' usage of the DSL tool compared with how many times they used aural support. This is presented in Figure 7.1 and it shows that the audio facility was used most at the beginning of the study, and usage decreased during the two academic Terms. Interestingly, there was an increase in its use during the exam revision period.

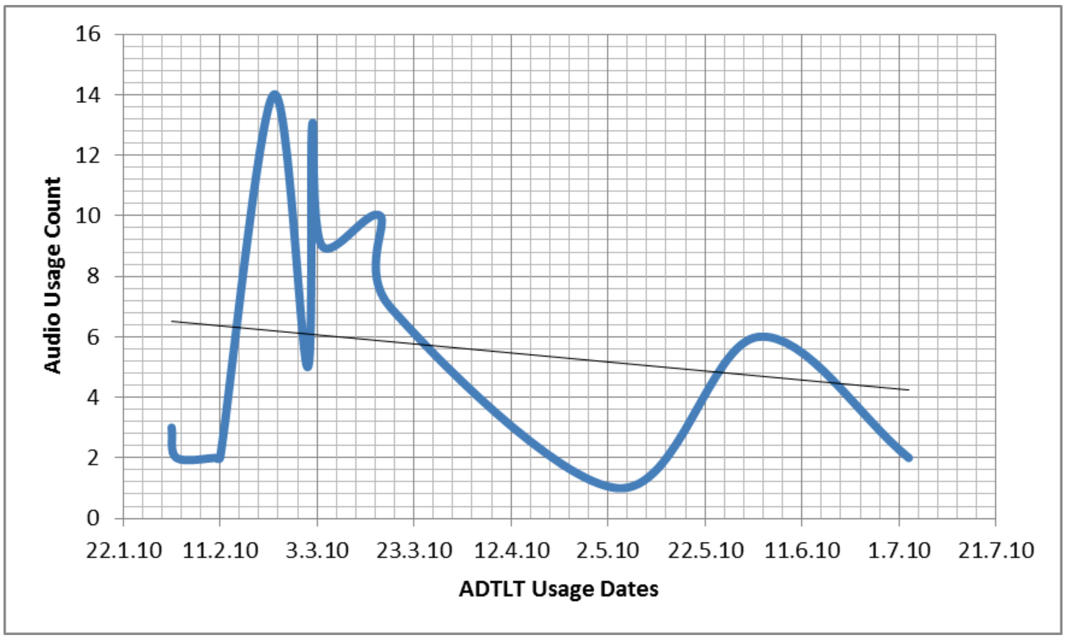


Figure 7.1: The use of audio during the experiment

The linear regression line in Figure 7.1 shows that students retreated from using aural instructions over the period of the experiment. The conclusion that can be drawn is that the audio facility was of most help to the students in the first part of the experimental period as they were learning the basic concepts, and that their use decreased as they became more interested in the visualisations. It is also possible that the DSL environment might have a design limitation that resulted in its failure to maintain students' interest in aural instructions throughout the course. However, the increased use of the audio facility before the exams suggests that students had not abandoned it as an unnecessary or unhelpful function but that they remembered its earlier support and hoped that it would aid their revision of data structure concepts.

The triangulation of the presented pieces of evidence from the data analysis indicates that the answer to research question Q5 is that the DSL environment does benefit students' learning experience regardless of the type of instructions used.

Thus, although no definite answer could be given to research question Q4, the answers to research questions Q3 and Q5 provide enough evidence to support the hypothesis H2 that "The use of aural instructions in teaching data structures to CS students has a positive effect on student perception of data structure concepts."

7.3.4 Threats

The first limitation of the validity of the presented evidence that supports the answers to the research questions in this section relates mainly to the automated collection of data about the use of aural instructions. The threat is that if a student chose to use aural instructions but did not use the headphones, it would mean that the student was only interested in the visualisations and not in the type of support instructions. Other threats concern the limitations of using questionnaire and interview results, especially when the data collected represents the views of only some of the students involved in the study.

7.4 Students' perceptions of the benefits of using the DSL tool

7.4.1 Introduction

This section represents an important focus of this thesis. It investigates the use of the DSL tool as a method to test the benefit of integrating aural instructions with visualisation in an interactive learning environment. It presents the third hypothesis of the thesis, that is "Students perceive a positive benefit to their learning by using the DSL tool." The purpose of this investigation is to gather evidence to answer the following research questions.

- Q6 Do CS students choose to use the DSL tool while studying data structure?
- Q7 Of the three data structure types used in this experiment, which do CS students select to explore through the DSL tool?
- Q8 Do CS students perceive benefits from using the DSL tool to build a mental model of data structures?

7.4.2 Research Methods

Two research methods were used to provide evidence to answer these three research questions. The automated data collected about students' usage of the tool provided strong evidence about the students' engagement with the tool, and how it was used. Feedback provided by the students at the end of each visualisation task was also valuable evidence of any benefits they perceived from using the tool.

7.4.3 Results and Discussion

To help to answer research question Q6, "Do CS students choose to use the DSL tool while studying data structures?" Section 6.4.1 highlighted the results of data captured by the DSL about how the students used the prototype tool. The first set of data provided an average mean of usage time of 39.42 minutes for all students, as shown in Figure 6.1. However, since the standard deviation is large at 32.1 minutes, the average cannot be generalised over the whole sample of participating students. This result shows that students varied widely in the time they spent using the DSL tool, and that usage time does not give a clear indication of how students engaged with the DSL approach.

Therefore, more investigation was needed to ascertain what students did, regardless of the time they spent on it. The second set of data measured the volume of students' engagement with the DSL tool as shown in Table 6.2. The duration of the engagement, their effective usage of the DSL tool (as explained in Section 6.2), and the students' task comments all lead to the conclusion that students intentionally chose to use the DSL environment as a source of information to assist them in learning the concepts of the data structures. The triangulation of the presented pieces of evidence from the data analysis indicates that the answer to research question Q6 is that the CS students have chosen to use the DSL tool while studying data structures.

To help to answer the research question Q7, that is "Of the three data structure types used in this experiment, which do CS students select to explore through the DSL tool?", Table 6.2 presented data about how many times students logged in to use each of the visualisations tasks in an effective way. The results show that students engaged with the approach and performed a high number of interactions with all the available types of visualisation tasks and that there are no significant differences between students' use of the DSL tool for each of the three visualisations. The data shows that most students engaged extensively and effectively with the DSL environment over the period it was available to them and that they were generally satisfied with its effectiveness in aiding their learning experience. However, the data does not provide information about what would have happened if students had not used the DSL environment at all, as there was no comparison against students' performance in previous years, or with a control group for reasons explained in Section 5.1.2. The presented pieces of evidence from the data analysis, however, indicates that there is no definite answer to research question Q7 about which data structure students select to explore the most through the DSL tool as the students have used all the data structures almost equally.

The help to answer the research question Q8, that is "Do CS students perceive benefits from using the DSL tool to build a mental model of data structures?", participants were asked in the end-of-experiment questionnaire whether the DSL environment had helped them in building a mental model of data structures. As discussed in Section 6.4, 93.8% of students reported that their use of the DSL tool helped them to create a mental model of the three data structures offered in the experiment. Moreover, in the end-of-experiment interviews, students were positive about the value of the environment, and reported that it had helped in reinforcing the concepts of the three data structures. So an answer to the research question Q8 would be that the DSL tool benefited students to build a mental model of data structures.

Thus, this thesis has demonstrated some support for hypothesis H3 that "Students perceive a positive benefit to their learning by using the DSL tool."

7.4.4 Threats

The main limitation of the presented evidence that supports the answers to the three research questions Q6, Q7 and Q8 is that no data is available that enables a comparison to be made between the students who used the DSL environment with those who did not. Even if this data were available, students who choose not to be engaged with the DSL tool might have had access to other forms of assistance to help them to understand the concepts of data structures. Moreover, the students who participated in the experiment might also have used other resources in conjunction with the DSL environment, so it is not possible entirely to isolate the effects of the DSL environment on students' understanding of the concepts of data structures.

7.5 Students' assessment marks and their engagement with the DSL environment

7.5.1 Introduction

This section examines the fourth hypothesis of the thesis, which is "There is a significant and positive relationship between a CS student's level of engagement with the DSL tool and his/her level of achievement, as measured by the official assessment marks." This involves a comparison of the amount of time each student spent using the DSL environment and his/her mark in the official assessment. It will also look at any evidence that enables this study to identify those students who most needed the DSL tool. The purpose of this investigation is to gather results to answer of the following research question.

- Q9 What is the level of achievement of CS students who choose to use the DSL tool the most?

7.5.2 Research Method

To investigate this hypothesis and answer the research question, two methods were used to gather the necessary information. First, the DSL tool's automated data collection gathered information about the duration of students' usage of the tool. Secondly, the researcher obtained access to the students' end-of-year assessment marks. The comparison between these two data sets looked for correlation between students' usage and their level of achievement. In addition, it looked to see whether conclusions could be drawn about the type of student who most needed the help provided by the DSL environment.

7.5.3 Results and Discussion

Section 6.4.5 investigated students' level of attainment in relation to the duration of their engagement with the DSL tool, and Figure 6.4 plotted a comparison of these two pieces of data for each participant in the experiment. This showed that, in general, the students who spent more time using the DSL tool obtained lower marks in the end-of-year assessment. This result suggests that less able students were keener to use the DSL environment than more able students. This does not mean that use of the DSL environment led to lower assessment marks; indeed, the low-achieving students might have performed even less well if they had not used the DSL tool. In any case, correlation does not imply causation, and there are many factors that could have affected the students' progress. The most likely explanation of the result is that less able students realise that they need more help to understand the assignment and, therefore, they will tend to use more of the learning resources available to them. Moreover, the lecturer who taught the data structure module feedback about the DSL environment clearly indicated that students have benefited from their engagement with this environment. The lecturer also expressed his interest in using the DSL environment for the next coming year.

The answer to research question Q9 is that it is the less able CS students who choose to use the DSL tool the most. This answer does not support hypothesis H4 that "There is a significant and positive relationship between a CS student's level of engagement with the DSL tool and his/her level of achievement, as measured by the official assessment marks" and the hypothesis proved to be not true.

7.5.4 Threats

The confound that could affect this result is the limitation of the design that did not specify pre and post tests to evaluate the students' level before and after their experience with the DSL environment. However, as explained in Section 5.1.2, pre and post test methodology was rejected because the study aimed to support students' learning over the duration of the academic year rather than to investigate the effects of a short intervention and, over that period, there were many other variables that could have affected students' understanding of the concepts of data structures.

7.6 Research Conclusions

7.6.1 Introduction

This final section summarises the conclusions of this study on the effect of the DSL environment on students' learning experience in the PDS module in the CS Department of Durham University. It reviews the answers to the research questions listed in Table 1.1 and highlights the contributions of this study to research into CS teaching and learning. It also highlights the limitations and weaknesses of the study and, finally, suggests some enhancements that could be applied in future research in this field.

7.6.2 Research contributions

The main research contribution of this study is to present a new environment to assist the teaching of novice CS students in the complicated but crucial concepts of programming. This approach suggests the use of aural instructions with focused visualisation to speed up the learning process at the start of the course and to benefit students in the long term. A prototype learning tool, called the DSL tool, was developed within an interactive learning environment to facilitate the implementation of the DSL approach to assist students in learning the concepts of data structures.

As well as assisting students, the DSL environment was designed to record rich usage data that enabled investigation and analysis of the students' learning experience while they were engaged with it. Valuable qualitative feedback was also obtained from the students.

7.6.3 Answers to the research questions and hypothesis

To meet the criteria for the success of this research and to reach worthwhile conclusions, it is important to answer the nine research questions identified in Table 1.1, and indicate whether or not the research hypotheses were or were not supported.

Q1. What existing research evidence is there that the simultaneous use of aural instruction along with visualisation will reduce cognitive load when learning data structures?

The study of cognitive load focused on the importance of understanding how student perceive knowledge. Cognitive load is the amount or measurement of mental effort needed by a student's brain to learn something, and this formed the base for designing the DSL environment, the aim of which was to keep the cognitive load to its minimum. Section 2.6 of this thesis presented a range of studies that investigated different approaches that could possibly provide answers to the research question Q1, and specifically the use of short instructions and their effect on students' response times. Section 7.2.3 discussed the connection between the use of response times as a measurement of cognitive load in the DSL environment and previous research in this field. Based on that relationship, a logical conclusion was drawn that the use of aural instruction with visualisation can reduce students' response times and that, in turn, the reduction of response times is a likely indicator of a reduction in their cognitive load.

Q2. Does the combination of aural instruction and visualisation reduce students' response time for task completion compared with textual instruction and visualisation?

The answer to this research question was presented separately in the mini study presented in Chapter 4. The mean of the response times for students in treatment group A, who had access to the DSL tool using textual instructions with the visualisation, was 12.60 seconds. The mean of the response times for students in treatment group B, who had access to the DSL tool using aural instruction with the visualisation, was 9.24 seconds. Table 4.4 showed that the difference between the response times of students who used audio only and those who used text only instructional methods was statistically significant. The conclusion of the study presented in Section 4.6 was that aural instructions result in significantly quicker responses compared to those achieved by textual instructions.

Thus, the answers to research questions Q1 and Q2 provided positive evidence that supports hypothesis H1 that "Reducing cognitive load improves student engagement and outcomes when learning data structures."

Q3. Do CS students perceive benefits from aural instructions along with visualisation when studying data structures?

As discussed in Section 7.3.2, the perception of benefits was measured from student responses to a questionnaire and interview questions. Section 6.4.4 presented students' responses to the questionnaire at the end of the academic year. These showed that they perceived that the DSL environment helped them to build mental models of data structures. A majority (93%) of the students who responded to the questionnaire reported that the aural instructions were clear and easy to follow. The results of the interviews, discussed in Section 6.5.2, confirmed this finding. However, some external factors may have resulted in reducing the choice of aural instructions over the textual instruction, such as the students' own learning styles, students forgetting to bring their headphones to the lab, and previous unfavourable experiences with audio based learning materials.

Q4. Is there a relationship between visualisation type and CS students' choice of instruction type?

Section 6.4.3 presented the analysis of the usage of the aural instruction format. Students who were actively engaged with the DSL tool used aural instructions in 54% of the tasks they performed. However, based on the results shown in Table 6.3, there was no significant difference between the use of the two types of instructions in each of the three the visualisation tasks. Consequently, as discussed in Section 7.3.3, no conclusion can be drawn about whether the difficulty level of the task has a significant impact on the students' choice of type of instructions.

Q5. Do students prefer the DSL environment based on visualisation only regardless the type of instruction?

To answer this question, a triangulation of different results was discussed in Section 7.3.3. Students' feedback about their usage of the DSL environment showed that they benefited from using aural instructions and that such instructions were very useful and interactive. The interviewees agreed that the aural instructions were clear and easy to understand. Figure 7.1 presented a timeline of students' usage of the DSL tool over the period of the experiment and this showed that the audio facility was used greatly at the beginning of the experiment, and that usage decreased over time. Interestingly, there was an increase in its use during the revision period. A conclusion can be drawn that the DSL environment is beneficial to learners, especially at the beginning of their course when they are being introduced to new concepts, and near the end of the course as an exam revision aid.

Thus, although no definite answer could be given to research question Q4, the answers to research questions Q3 and Q5 provide enough positive evidence to support hypothesis H2 that "The use of aural instructions in teaching data structures to CS students has a positive effect on student perception of data structure concepts."

Q6. Do CS students choose to use the DSL tool while studying data structures?

The results of data captured by the DSL environment about how the students used the prototype tool were discussed in Section 6.4.1. The first set of data provided the average mean of usage times in minutes for all students, as shown in Figure 6.1. The average usage time was 39.42 minutes. The second set of data measured the amount of students' engagement with the DSL tool as shown in Table 6.2. Based on this usage data about engagement duration and effective usage of the DSL environment, and on students' feedback about their experience, a conclusion can be drawn. This is that students intentionally chose to use the DSL environment as a source of information to assist them in learning the concepts of the three data structures offered in this study.

Q7. Of the three data structure types used in this experiment, which do CS students select to explore through the DSL tool?

Section 7.4.3 discussed how students engaged with the DSL approach and performed a high number of interactions with all three visualisation tasks: BTT, LL and BST. Although there are no statistically significant differences between the use of the DSL tool with each of the three visualisations, Figure 6.2 shows that the BTT task, usually considered to be the most complex of the three, had the most interactions from students. In addition, BTT received the highest student rating (mean of 3.8 out of 5) among the three investigated tasks. In questionnaire and interview answers, students suggested developing the DSL tool to include help with understanding the concepts of the self-balancing binary search tree (AVL Tree) and graphs.

Q8. Do CS students perceive benefits from using the DSL tool to build a mental model of Data Structures?

As discussed in Section 7.4.3, the questionnaire results showed that 93.8% of the students reported that the DSL environment helped them to build a mental model of data structures. The students who were interviewed were also positive about the value of the DSL environment, and reported that it had helped them in reinforcing the concepts of the three types of data structures.

Based on the answers to the research questions Q6, Q7, and Q8 hypothesis H3 that "Students perceive a positive benefit to their learning by using the DSL tool" is proved.

Q9. What is the level of achievement of CS students who choose to use the DSL tool the most?

Section 7.5.3 of this thesis presented the relation between student attainment in the end of the year assessment exam and engagement with the DLS environment. As shown in Figure 6.4, students who gained lower marks in the assessment spent a longer time engaging with the DSL environment. This result indicates that less able students were keener than others to use the DSL environment. This may have been because they realised they needed more help to gain the required understanding of the concepts of data structures. This outcome, therefore, does not support hypothesis H4 that “There is a significant and positive relationship between a CS student’s level of engagement with the DSL tool and his/her level of achievement, as measured by the official assessment marks.”

7.6.4 Limitations of the research

This research has limitations. All the limitations and threats were discussed individually in Sections 7.2.4, 7.3.4, 7.4.4, and 7.5.4. As with any study that investigates students’ learning through data collection and observation, there are many environmental and technical factors that can affect the variables of an experiment. The environmental factors relate to the students’ personal experience, learning styles and willingness to seek knowledge from outside resources, as well as the availability of those resources. The technical factors usually relate to the limited usability of any prototype tool developed for the research, the limited functionalities available to students and the lack of training resources. The methodologies used can also cause limitations to confidence in the research results.

The first threat is related to the validity of the resources used for the systematic literature review. No study was found that claimed that a reduction of CS students' cognitive load can be accomplished by using short aural instructions with visualisation in a multimedia learning environment. However, this approach was used in the first, mini experiment due to the lack of prior work that documents a common and firm methodology for measuring cognitive load. The discussion on cognitive load in Section 2.2.8 suggests that the methodology used has a face validity and this rendered it the most suitable method for the purpose of the first experiment.

Threats to the validity of the results of the second experiment should also be considered, especially if they could have a direct or indirect effect on the results of this research. First, the questionnaire had a comparatively low response rate so the collected data may not be representative of the experiment's participants. Also, some technical faults in the prototype tool may have caused some students to be dissatisfied at an early stage with the experience of engaging with the DSL approach and this may have been responsible for reducing the number of active participants and, thus, for reducing the amount and quality of the data about the effect of receiving aural instructions on students' learning experience.

The results of this study could be compromised by the limited use by students of interactive aural instruction. This was caused by the limited time available to implement an intelligent and comprehensive interactive environment and the limited resources available to introduce the approach to the students and explain its potential benefits. Having more time and resources could definitely help in overcoming such issues. Use of the DSL environment was also entirely voluntary and some students may have seen participating in a research experiment as an unwanted extra burden in a busy work schedule.

Finally, the validity of the research results can be threatened by the lack of a fully controlled experiment on different student groups with a pre and post-test to evaluate what the group with access to the DSL approach actually achieved compared to the achievements of the group without access to it.

7.7 Future Work

Section 7.6 pointed out some possibilities for improvements to the DSL approach that might assist other researchers in this field. While the DSL environment proved to be useful to students and the successful results support the use of aural instructions with interactive visualisation, there were some technical limitations that could be removed by future work. For example, adding visualisations that help students in learning AVL Trees and Graphs algorithms would undoubtedly improve its usefulness. These additions were recommended by the students themselves in comments they made while using the DSL approach as well as later in answers to questionnaires and in interviews.

The DSL tool design is easy to understand, and it is flexible and easy to manipulate, so the tool could be easily extended by anyone with a good knowledge of a programming language and the concepts of data structures. The following technical improvements could be applied to enhance the performance of the prototype tool:

- The current instructions list is limited and for experimental use only. A wider instructions database could be created so that students can engage more actively in the learning tasks. The instruction database could be used as a source of both textual and aural instructions.

- Implementing the visualisation of AVLTrees and Graphs by using the node structure shown in Figure 3.5, along with the animating methods described in Section 3.6.
- Breaking down the addition and deletion animation of Linked Lists and Binary Trees into more detailed steps so that the student can have a better understanding of each step of the visualisation. The current prototype tool shows the student only the final result of the addition or deletion of nodes, although the generation of these results implemented through the detailed route is used by the algorithms of the data structures.

The following suggestions would improve the scope and methodology for future research on this topic.

- This research could be extended by including a comparative study of students who experience the environment and those who choose not to participate. This would offer a deeper understanding of how the DSL environment affects students' learning experience.

- Multiple cross-sectional studies could assess each aspect of the DSL approach individually in shorter experiments instead of conducting a longitudinal study.

- Investigating the impact of students' learning styles on the choice of instructional method. It is highly recommended that an assessment of students learning styles should be performed before introducing them to the DSL learning environment.

- Conducting pre and post-tests to evaluate students' level of achievement before and after using the DSL environment.

- Using an eye tracking methodology to test the usability of the environment and the impact on students' cognitive load.

Finally, it is highly recommended that the DSL environment should be fully integrated with the PDS module content, so that it becomes an extra teaching resource to help both lecturers and students in the process of teaching and learning data structures, as well as providing a resource for approaching the concepts of other programming languages.

7.8 Final Conclusion

In summary, this thesis studied the effectiveness of using aural instructions with visualisation in an interactive learning environment (the DSL environment). It concentrated on the use of this approach when teaching novice computer science students the concepts of data structures. This approach aimed to reduce the cognitive load on students' working memory by exploiting the dual sensory channels for information processing. This concept was discussed in Section 2.6.1.1 and illustrated by Mayer's multimedia learning model in Figure 2.13.

A triangulation method was employed to explore the relationship between the results obtained by the qualitative and quantitative research methods described in Chapter 5. The results, discussed in chapters 6 and 7, showed the benefits of using this research environment on students' learning experience. Data was collected by using a prototype tool created for this purpose, by testing students' perception of this approach through an end of term questionnaire and by conducting a group interview with a sample of students. A full discussion of the results was presented in Chapter 7.

The DSL environment was created to test the four hypotheses of this thesis and to answer the nine research questions listed in Table 1.1. The implementation of the

DSL environment provided enough evidence of the value of aural instructions to show that this is a research channel that is worth pursuing and that its development would extend its benefits. The research also demonstrated a substantial interest from the students and the lecturer in continuing to use the DSL environment as an extra resource in learning the concepts of data structures.

Bibliography

(2010) Faculty Handbook Online. IN Science, D. O. C. (Ed.) Durham, Durham University.

Ala-Mutka, K. Problems in Learning and Teaching Programming. Finland, Institute of Software Systems, Tampere University of Technology, Finland.

Aleksandra, K.-M., Boban, V., Mirjana, I. & Zoran, B. (2010) Integration of Recommendations and Adaptive Hypermedia into Java Tutoring System. *Computer Science and Information Systems*.

Alfonseca, E., Carro, R. M., Martín, E., Ortigosa, A. & Paredes1, P. (2006) The Impact of Learning Styles on Student Grouping for Collaborative Learning. *User Modeling and User-Adapted Interaction,* 16, 377-401.

Allen, J. & Terman, C. J. (1999) An Interactive Learning Environment for Vlsi Design. *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on.*

Alty, J. L., Rigas, D. & Vickers, P. (1997) Using Music as a Communication Medium. *CHI '97 extended abstracts on Human factors in computing systems: looking to the future.* Atlanta, Georgia, ACM New York, NY, USA.

Ando, M. & Ueno, M. (2008) Cognitive Load Reduction on Multimedia E-Learning Materials. *Advanced Learning Technologies, 2008. ICALT '08. Eighth IEEE International Conference on.*

Andrian, M., Louis, F. & Jonathan, I. M. (2003) 3d Representations for Software Visualization. *Proceedings of the 2003 ACM symposium on Software visualization.* San Diego, California, ACM.

Arnold, P., Stephen, S., Lauri, M., Linda, M., Elizabeth, A., Jens, B., Marie, D. & James, P. (2007) A Survey of Literature on the Teaching of Introductory Programming. ACM.

Arons, B. & Mynatt, E. (1994) The Future of Speech and Audio in the Interface: A Chi '94 Workshop. SIGCHI Bulletin.

Atherton, J. S. (2005) Learning and Teaching: Deep and Surface Learning.

Atkinson, R. K. (2002) Optimizing Learning from Examples Using Animated Pedagogical Agents. *Journal of Educational Psychology,* 94, 416-427.

Ball, G. & Breese, J. (1998) Emotion and Personality in a Conversational Character. *Proceedings of WECC '98: The First Workshop on Embodied Conversational Characters.*

Barbara, M., Deborah, L. & Stephen, C. (2004) Evaluating the Effectiveness of a New Instructional Approach. *SIGCSE Bull.,* 36, 75-79.

Barchéus, F. (2006) Time Analysis of Atc Radio Traffic in a Simulated Free Flight Scenario. *Swedish Human Factors Network (HFN) conference.* Linköping, Sweden, Swedish Network for Human Factors (HFN).

Beck, J. E. (2005) Engagement Tracing: Using Response Times to Model Student Disengagement. *Proceeding of the 2005 conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology.* IOS Press.

Bednarik, R., Moreno, A. & Myller, N. (2006) Various Utilizations of an Open-Source Program Visualization Tool, Jeliot 3. *Informatics in Education,* 5, 195-206.

Behringer, R., Behringer, R., Chen, S., Sundareswaran, V., Wang, K. A. W. K. & Vassiliou, M. A. V. M. (1999) A Novel Interface for Device Diagnostics Using Speech Recognition, Augmented Reality Visualization, and 3d Audio Auralization. IN Chen, S. (Ed.) *Multimedia Computing and Systems, 1999. IEEE International Conference on.*

Bertram, M., Bertram, M., Deines, E., Mohring, J., Jegorovs, J. A. J. J. & Hagen, H. A. H. H. (2005) Phonon Tracing for Auralization and Visualization of Sound

Phonon Tracing for Auralization and Visualization of Sound. IN Deines, E. (Ed.) *Visualization, 2005. VIS 05. IEEE.*

Biggs, J. B. (2003) *Teaching for Quality Learning at University: What the Student Does*, Society for Research into Higher Education.

Bloom, B. S. & Krathwohl, D. R. (1956) *Taxonomy of Educational Objectives: The Classification of Educational Goals. Handbook I: Cognitive Domain*, Longmans.

Boroni, C. M., Goosey, F. W., Grinder, M. T. & Ross, R. J. (1998) A Paradigm Shift! The Internet, the Web, Browsers, Java and the Future of Computer Science Education. *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education.* Atlanta, Georgia, United States, ACM Press.

Briana Lowe, W., James, D. & Monica, A. (2009) Alice and Robotics in Introductory Cs Courses. *The Fifth Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations.* Portland, Oregon, ACM.

Brosgol, B. M. (1998) A Comparison of Ada and Java as a Foundation Teaching Language. *ACM SIGAda Ada Letters,* XVIII, 12-38.

Brown, M. H. & Hershberger, J. (1992) Color and Sound in Algorithm Animation. *Computer,* 25, 52-63.

Brown, M. H. & Sedgewick, R. (1984) A System for Algorithm Animation. ACM.

Brünken, R., Plass, J. L. & Leutner, D. (2004) Assessment of Cognitive Load in Multimedia Learning with Dual-Task Methodology: Auditory Load and Modality Effects. *Instructional Science,* 32, 115-132.

Byrne, M. D., Catrambone, R. & Stasko, J. T. (1996) Do Algorithm Animations Aid Learning? *Technical ReportGITGVU9618.*

Carver, C. A., Jr., Howard, R. A. & Lane, W. D. (1999) Enhancing Student Learning through Hypermedia Courseware and Incorporation of Student Learning Styles. *Education, IEEE Transactions on,* 42, 33-38.

Chamillard, A. T. & Sward, R. E. (2005) Learning Styles across the Curriculum. *SIGCSE conference on Innovation and technology in computer science education.* Caparica, Portugal, ACM Press.

Clark, D. (1995) Kolb's Learning Styles.

Clark, R. E. (2001) What Is Next in the Media and Methods Debate. IN Clark, R. E. (Ed.) *Learning from Media: Arguments, Analysis, and Evidence.* Information Age Pub.

Coffield, F., Moseley, D., Hall, E. & Ecclestone, K. (2004) Learning Styles and Pedagogy in Post-16 Learning. A Systematic and Critical Review. London, Learning and Skills Research Centre.

Cooper, G. (1998) Research into Cognitive Load Theory and Instructional Design at Unsw. Sydney, University of New South Wales, Australia.

Cross, J. H. & Hendrix, D. (2006) Workshop Jgrasp: An Integrated Development Environment with Visualizations for Teaching Java in Cs1, Cs2, and Beyond. *Frontiers in Education Conference, 36th Annual.* San Diego, CA

Cross, J. H., Hendrix, D. & Umphress, D. A. (2004) Jgrasp: An Integrated Development Environment with Visualizations for Teaching Java in Cs1, Cs2, and Beyond. *Frontiers in Education, 2004. FIE 2004. 34th Annual.*

Culwin, F., Adeboye, K. & Campbell, P. (2006) Pooples - Pre-Object Oriented Programming Learning. Environments. *Proceedings of 7th Annual Conference of the ICS HE Academy.* Trinity College, Dublin, Higher Education Academy.

Daly, C. & Horgan, J. M. (2004) An Automated Learning System for Java Programming. *IEEE Transactions on Education,* 47, 10 - 17.

David, F., Thomas, N. & Jason, W. (2008) Sorting out Sorting: The Sequel. *SIGCSE Bull.,* 40, 174-178.

Dawson, S., McWilliam, E. & Tan, J. P.-L. (2008) Teaching Smarter: How Mining Ict Data Can Inform and Improve Learning and Teaching Practice. *Hello! Where are you in the landscape of educational technology?* , Proceedings ascilite Melbourne 2008.

Decker, R. & Hirshfield, S. (1995) The Top 10 Reasons Why Oop Can't Be Taught in Cs1. *3C ON-LINE,* 2, 7-13.

Dennis, H. (2000) Speech-Enabling the Data Structures Curriculum. 15, 130-141.

Depradine, C. & Gay, G. (2004) Active Participation of Integrated Development Environments in the Teaching of Object-Oriented Programming. *Computers & Education,* 43, 291-298.

Deubel, P. (2003) An Investigation of Behaviorist and Cognitive Approaches to Instructional Multimedia Design. *Journal of Educational Multimedia and Hypermedia,* 12, 63-90.

Dexter, S. (2000) Teaching Applet Programming to Non-Majors - Virtually. *Frontiers in Education Conference, 2000. FIE 2000. 30th Annual.*

Dino, S. & Wayne, B. (2007) Interactive Visualization for the Active Learning Classroom. ACM.

Dougherty, J. P. (2008) Using Lyrics and Music to Reinforce Concepts. *Journal of Computing Sciences in Colleges,* 23, 106 - 113

Dror, I. E. & Schmitz-Williams, I. C. (2005) Older Adults Use Mental Representations That Reduce Cognitive Load: Mental Rotation Utilizes Holistic Representations and Processing. *Experimental Aging Research: An International Journal Devoted to the Scientific Study of the Aging Process,* 31, 409-420.

Dunn, R. (1990) Understanding the Dunn and Dunn Learning Styles Model and the Need for Individual Diagnosis and Prescription. *Reading &amp; Writing Quarterly,* 6, 223 - 247.

Durant, F. (2007) Sea, Speech and Sun. Cannes, France, Nuance Conversations.

Ehlert, A. & Schulte, C. (2009) Empirical Comparison of Objects-First and Objects-Later. *Proceedings of the fifth international workshop on Computing education research workshop.* Berkeley, CA, USA, ACM.

Evans, C. & Gibbons, N. J. (2007) The Interactivity Effect in Multimedia Learning. *Computers & Education,* 49, 1147-1160.

Felder, R. M. (1988) How Students Learn: Adapting Teaching Styles to Learning Styles. *Frontiers in Education Conference, 1988., Proceedings.*

Felder, R. M. & Soloman, B. A. (1988) Learning Styles and Strategies.

Felder, R. M. & Soloman, B. A. (1988) Learning Styles and Strategies.

Franklin, J. A. (2001) Computer Generated Music as a Teaching Aid for First Year Computing. Consortium for Computing Sciences in Colleges.

Gaver, W. W., Smith, R. B. & O'Shea, T. (1991) Effective Sounds in Complex Systems: The Arkola Simulation. *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology.* New Orleans, Louisiana, United States, ACM.

Gerjets, P. & Kirschner, P. (2009) Learning from Multimedia and Hypermedia. IN Balacheff, N., Ludvigsen, S., Jong, T., Lazonder, A. & Barnes, S. (Eds.) *Technology-Enhanced Learning.* Springer Netherlands.

Gilbert, J. E. & Han, C. Y. (1999) Adapting Instruction in Search of [`]a Significant Difference'. *Journal of Network and Computer Applications,* 22, 149-160.

Goldstein, C., Leisten, S., Stark, K. & Tickle, A. (2005) Using a Network Simulation Tool to Engage Students in Active Learning Enhances Their Understanding of Complex Data Communications Concepts. *Proceedings of the 7th Australasian conference on Computing education - Volume 42* Newcastle, New South Wales, Australia Australian Computer Society, Inc.

Graf, S., Lin, T., Jeffrey, L. & Kinshuk (2006) An Exploratory Study of the Relationship between Learning Styles and Cognitive Traits. *Innovative Approaches for Learning and Knowledge Sharing.* Berlin, Heidelberg, Springer

Grigoriadou, M., Papanikolaou, K., Kornilakis, H. & Magoulas, G. (2001) Inspire: An Intelligent System for Personalized Instruction in a Remote Environment. *Proceedings of 3rd Workshop on Adaptive Hypertext and Hypermedia.* Sonthofen, Germany.

Grissom, S., McNally, M. F. & Naps, T. (2003) Algorithm Visualization in Cs Education: Comparing Levels of Student Engagement. *Proceedings of the 2003 ACM symposium on Software visualization* San Diego, California, ACM Press.

Gross, P. & Powers, K. (2005) Evaluating Assessments of Novice Programming Environments. *Proceedings of the first international workshop on Computing education research.* Seattle, WA, USA, ACM.

Gurka, J. S. & Citrin, W. (1996) Testing Effectiveness of Algorithm Animation. *Visual Languages, 1996. Proceedings., IEEE Symposium on.*

Guzdial, M., Santos, P., Badre, A., Hudson, S. & Gray, M. (1994) Analyzing and Visualizing Log Files: A Computational Science of Usability. *IEEE Transactions on Reliability*.

Hagan, D. & Markham, S. (2000) Teaching Java with the Bluej Environment. *ASCILITE 2000.* Coffs Harbour, Australia, ASCILITE

Hamer, J. (2004) Visualising Java Data Structures as Graphs. *Proceedings of the sixth conference on Australasian computing education - Volume 30.* Dunedin, New Zealand Australian Computer Society, Inc.

Hamid, S. H. A. & Chuan, T. H. (2006) Designing Learning Styles Application of E-Learning System Using Learning Objects. *Advances in Web Based Learning – Icwl 2006.* Berlin, Heidelberg, Springer

Hamilton-Taylor, A. G. & Kraemer, E. (2002) Ska: Supporting Algorithm and Data Structure Discussion. *Proceedings of the 33rd SIGCSE technical symposium on Computer science education.* Cincinnati, Kentucky, ACM New York, NY, USA.

Hanley, M. (2010) A Media Model for E-Learning Content: Project Lifecycle 4. *E-Learning Curve Blog.*

Hansen, S., Schrimpsher, D. & Narayanan, N. H. (1998) From Algorithm Animations to Animation-Embedded Hypermedia Visualizations.Pdf. IN Department of Computer Science & Engineering, A. U. (Ed.) *Visual Information, Intelligence & Interaction Research Group.* Auburn,.

Hensler, B. & Beck, J. (2006) Better Student Assessing by Finding Difficulty Factors in a Fully Automated Comprehension Measure. *Intelligent Tutoring Systems.* Springer Berlin / Heidelberg.

Houghton, W. (2004) Learning and Teaching Theory for Engineering Academics. The Higher Education Academy Engineering Subject Centre.

Howard, R. A., Carver, C. A. & Lane, W. D. (1996) Felder's Learning Styles, Bloom's Taxonomy, and the Kolb Learning Cycle: Tying It All Together in the Cs2 Course. *the twenty-seventh SIGCSE technical symposium on Computer science education.* Philadelphia, Pennsylvania, United States ACM Press.

Hu, P. J. H., Hui, W., Clark, T. H. K. & Tam, K. Y. (2007) Technology-Assisted Learning and Learning Style: A Longitudinal Field Experiment. *Systems, Man and Cybernetics, Part A, IEEE Transactions on,* 37, 1099-1112.

Hubscher-Younger, T. & Narayanan, N. H. (2001) How Undergraduate Students' Learning Strategy and Culture Effects Algorithm Animation Use and Interpretation.

Hübscher-Younger, T. & Narayanan, N. H. (2003) Dancing Hamsters and Marble Statues: Characterizing Student Visualizations of Algorithms. *Proceedings of the 2003 ACM symposium on Software visualization.* San Diego, California, ACM Press.

Iyengar, S. S., Pangburn, B. E. & Mathews, R. C. (2000) Web-Based Multimedia Development Techniques for the Instruction of Abstract Concepts in Computer Science. *Multimedia Software Engineering, 2000. Proceedings. International Symposium on.*

Jain, J., Cross, J. H., Hendrix, T. D. & Barowski, L. A. (2006) Experimental Evaluation of Animated-Verifying Object Viewers for Java. *Proceedings of the 2006 ACM symposium on Software visualization.* Brighton, United Kingdom, ACM.

Jarc, D. J. & Feldman, M. B. (1998) An Empirical Study of Web-Based Algorithm Animation Courseware in an Ada Data Structure Course.

Annual International Conference on Ada Washington, D.C., United States ACM Press.

Jerding, D. F. & Stasko, J. T. (1994) Using Visualization to Foster Object-Orianted Program Understanding. Atlant, Georgia Institute of Technology.

Jyothi, S., McAvinia, C. & Keating, J. G. (2007) An Interaction Visualisation Tool for a Learning Management System *of the 2007 conference of the center for advanced studies on Collaborative research.* Richmond Hill, Ontario, Canada, ACM   New York, NY, USA.

Kahneman, D. (1973) *Attention and Effort*, Prentice-Hall.

Kalyuga, S. (2006) Expert-Novice Differences and Adaptive Multimedia. *Digital Multimedia Perception and Design.* IGI Publishing.

Karavirta, V., Korhonen, A., Malmi, L. & Stalnacke, K. (2004) Matrixpro - a Tool for Demonstrating Data Structures and Algorithms Ex Tempore. *Advanced Learning Technologies, 2004. Proceedings. IEEE International Conference on.*

Kazi, I. H., Jose, D. P., Ben-Hamida, B., Hescott, C. J., Kwok, C., Konstan, J. A., Lilja, D. J. & Yew, P. C. (2000) Javiz: A Client/Server Java Profiling Tool. *IBM Systems Journal,* 39, 96-117.

Kearney, M. (2004) Classroom Use of Multimedia-Supported Predict–Observe–Explain Tasks in a Social Constructivist Learning Environment. *Research in Science Education,* 34, 427-453.

Kehoe, A. & Pitt, I. (2006) Designing Help Topics for Use with Text-to-Speech. *Proceedings of the 24th annual ACM international conference on Design of communication.* Myrtle Beach, SC, USA, ACM.

Khaled, R., Luxton, A. M., Noble, J., Ferres, L., Brown, J. & Biddle, R. (2004) Visualisation for Learning Oop, Using Aop and Eclipse. *Conference on Object Oriented Programming Systems Languages and Applications.* Vancouver, BC, CANADA, ACM   New York, NY, USA.

Khawaja, N. G. & Dempsey, J. (2007) Psychological Distress in International University Students: An Australian Study. Australian Academic Press.

Khuri, S. (2001) A User-Centred Approach for Designing Algorithm Visualizations. *Informatik/Informatique, Special Issue on Visualization of Software,* 2, 12-16.

Klob, D. (2005) Kolb Learning Styles, Kolb's Learning Styles Model and Experiential Learning Theory (Elt). www.businessballs.com.

Kolb, D. A. (1984) *Experiential Learning: Experience as the Source of Learning and Development*, Prentice-Hall.

Kolb, D. A., Boyatzis, R. & Mainemelis, C. (2001) *Perspectives on Thinking, Learning, and Cognitive Styles*, L. Erlbaum Associates.

Kolling, M., Koch, B. & Rosenberg, J. (1995) Requirements for a First Year Object-Oriented Teaching Language *SIGCSE technical symposium on Computer science education* Nashville, Tennessee, United States ACM New York, NY, USA

Kropp, W. & Blomqvist, J. (2001) Auralisation and Visualisation as Tools for Learning in Acoustics. *CAL-laborate.* UniServe Connections.

Lai, J. (2001) When Computers Speak, Hear, and Understand. ACM.

Larkin, T. & Budny, D. (2005) Learning Styles in the Classroom: Approaches to Enhance Student Motivation and Learning. *Information Technology Based Higher Education and Training, 2005. ITHET 2005. 6th International Conference on.*

Larkin, T. L., Feldgen, M. & Clua, O. (2002) A Global Approach to Learning Styles. *Frontiers in Education, 2002. FIE 2002. 32nd Annual.*

Longmire, W. (2000) A Primer on Learning Objects. IN Circuits, A. L. (Ed.), ASTD Learning Circuits.

Lowe, R. (1999) Extracting Information from an Animation During Complex Visual Learning. *European Journal of Psychology of Education,* 14, 225-244.

Maravic Cisar, S., Pinter, R., Radosav, D. & Cisar, P. Software Visualization: The Educational Tool to Enhance Student Learning. *MIPRO, 2010 Proceedings of the 33rd International Convention.*

Marton, F. & Saljo, R. (1976) On Qualitative Differences in Learning: I. Outcome and Process. *British Journal of Educational Psychology,* 46, 4-11.

Mayer, R. E. (2001) *Multimedia Learning*, Cambridge University Press.

Mayer, R. E. & Moreno, R. (2002) Aids to Computer-Based Multimedia Learning. *Learning and Instruction,* 12, 107-119.

Mayes, J. T. & Fowler, C. J. (1999) Learning Technology and Usability: A Framework for Understanding Courseware. *Interacting with Computers,* 11, 485-497.

McAllister, W. (2008) *Data Structures and Algorithms Using Java*, Jones and Bartlett Publishers.

McKinney, K. (2007) Active Learning.

McManus, D. A. (2001) The Two Paradigms of Education and the Peer Review of Teaching. *NAGT Journal of Geoscience Education,* 49, 423-434.

Messner, J. I. & Horman, M. J. (2003) Using Advanced Visualization Tools to Improve Construction Education. *CONVR 2003.* Virginia Tech.

Meyers, C. & Jones, T. B. (1993) *Promoting Active Learning: Strategies for the College Classroom*, Jossey-Bass.

Mia, S. & Beverly Park, W. (2000) Adaptive Content in an Online Lecture System. *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems.* Springer-Verlag.

Michael, F. M. (2002) Spoken Dialogue Technology: Enabling the Conversational User Interface. ACM.

Moloney, J., Moloney, J. & Harvey, L. (2004) Visualization and 'Auralization' of Architectural Design in a Game Engine Based Collaborative Virtual Environment

Visualization and 'Auralization' of Architectural Design in a Game Engine Based Collaborative Virtual Environment. IN Harvey, L. (Ed.) *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on.*

Moons, J. & De Backer, C. (2009) Rationale Behind the Design of the Eduvisor Software Visualization Component. *Electronic Notes in Theoretical Computer Science,* 224, 57-65.

Moor, B. D. & Deek, F. P. (2006) On the Design and Development of Uml-Based Visual Environment for Novice Programmers *1998 International Conference on Software Engineering: Education & Practice.*

More, A. J. (1989) Native Indian Learning Styles: A Review for Researchers and Teachers. *Journal of American Indian Education,* (Special Edition).

Moreno, A., Myller, N., Sutinen, E. & Ben-Ari, M. (2004) Visualizing Programs with Jeliot 3. *Proceedings of the working conference on Advanced visual interfaces.* Gallipoli, Italy, ACM.

Müldner, T., Mudner, T. & Shakshuki, E. (2004) A New Approach to Learning Algorithms

a New Approach to Learning Algorithms. IN Shakshuki, E. (Ed.) *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on.*

Müldner, T., Shakshuki, E. & Kerren, A. (2005) Using Structured Hypermedia to Explain Algorithms. *Proceedings of the 3rd IADIS International Conference e-Society'05*.

Najjar, J., Duval, E. & Wolpers, M. (2006) Towards Effective Usage-Based Learning Applications: Track and Learn from User Experience(S). *Advanced Learning Technologies, 2006. Sixth International Conference on.*

Naps, T. L. (2002) Exploring the Role of Visualization and Engagement in Computer Science Education. *Annual Joint Conference Integrating Technology into Computer Science Education* Aarhus, Denmark, ACM Press.

Navon, D. (1984) Resources-a Theoretical Soup Stone? *Psychological Review,* 91, 216-234.

Nicholson, D., Hamilton, D. & McFarland, D. (2007) Leveraging Learning Styles to Improve Student Learning: The Interactive Learning Model and Learning Combination Inventory. *Consortium for Computing Sciences in Colleges,* 22, 8-17.

Nicolaou, C. T., Nicolaidou, I. & Constantinou, C. (2003) The E-Learning Movement as a Process of Quality Improvement in Education. *Sixth International Conference on Computer Based Learning in Science (CBLIS 2003).* Nicosia, Cyprus.

Niemi, H. (2002) Active Learning--a Cultural Change Needed in Teacher Education and Schools. *Teaching and Teacher Education,* 18, 763-780.

Notess, M. & Neal, L. (2006) Deep Thoughts: Do Mandatory Online Activities Help Students Leave Surface-Learning Behind? *eLearn Magazine.*

Nourie, D. (2008) Young Developer Learning Path. Java.net.

Nycz, M. & Cohen, E. (2007) Basics for Understanding E-Learning. *Principles of Effective Online Teaching.* Santa Rosa, California, Informing Science Press.

Oates, B. J. (2006) *Researching Information Systems and Computing*, SAGE.

Olthoff, W. G. (1986) Augmentation of Object-Oriented Programming by Concepts of Abstract Data Type Theory: The Modpascal Experience. *onference proceedings on Object-oriented programming systems, languages and applications.* Portland, Oregon, United States, ACM.

Owen, G. S. (1999) Hypervis - Teaching Scientific Visualization Using Hypermedia.

Paredes, P. & Rodriguez, P. (2006) Considering Sensing-Intuitive Dimension to Exposition-Exemplification in Adaptive Sequencing. *Adaptive Hypermedia and Adaptive Web-Based Systems.* Springer Berlin / Heidelberg.

Pargas, R. P. & Goodbar, A. (2006) Work in Progress: Developing Tablet Pc Animations for Computer Science Courses. *Frontiers in Education Conference, 36th Annual.*

Parker, B. & Mitchell, I. (2006) Effective Methods for Learning: A Study in Visualization. *Journal of Computing Sciences in Colleges,* 22, 176 - 182

Pierson, W. C. & Rodger, S. H. (1998) Web-Based Animation of Data Structures Using Jawaa. 30, 267-271.

Popescu, E., Trigano, P. & Badica, C. (2007) Towards a Unified Learning Style Model in Adaptive Educational Systems. *Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on.*

Qiu, L. & Benbasat, I. (2005) An Investigation into the Effects of Text-to-Speech Voice and 3d Avatars on the Perception of Presence and Flow of Live Help in Electronic Commerce. *ACM Transactions on Computer-Human Interaction (TOCHI),* 12, 329-355.

Rai, S., Wong, K. W. & Cole, P. (2006) Game Construction as a Learning Tool. *Proceedings of the 2006 international conference on Game research and development.* Perth, Australia Murdoch University, Australia.

Rajala, T., Laakso, M.-j., Kaila, E. & Salakoski, T. (2008) Innovations in Practice Effectiveness of Program Visualization: A Case Study with the Ville Tool. *Journal of Information Technology Education: Innovations in Practice,* Volume 7, 15-32.

Rajaravivarma, R. & Pevac, I. (2003) When to Introduce Objects in Teaching Java. *System Theory, 2003. Proceedings of the 35th Southeastern Symposium on.*

Rajasingham, L. (2010) Will Mobile Learning Bring a Paradigm Shift in Higher Education? *Education Research International.*

Ramsden, P., Beswick, D. & Bowden, J. (1989) Effects of Learning Skills Intervention on First Year Students' Learning. *Human Learnin,* 5, 151-164.

Richards, J., Barowy, W. & Levin2, D. (1992) Computer Simulations in the Science Classroom. *Journal of Science Education and Technology,* 1, 67-79.

Roberts, A. (2001) Abc of Learning: Words Used to Make Theories About How We Learn.

Rogers, Y. (2008) A Comparison of How Animation Has Been Used to Support Formal, Informal, and Playful Learning. *Learning with Animation: Research Implications for Design.* Cambridge University Press.

Rosati, P., Dean, R. K. & Rodman, S. M. (1988) A Study of the Relationship between Students' Learning Styles and Instructors' Lecture Styles. *Education, IEEE Transactions on,* 31, 208-212.

Ruiz, M. d. P. P., Barriales, S. O., Pérez, J. R. P. & Rodríguez, M. G. (2003) Feijoo.Net- an Approach to Personalized E-Learning Using Learning Styles. *Lecture Notes in Computer Science.* Berlin / Heidelberg, Springer

Salmon, G. (2002) *E-Tivities: The Key to Active Online Learning*, Kogan Page.

Sánchez, J. (2007) A Model to Design Interactive Learning Environments for Children with Visual Disabilities. *Education and Information Technologies,* 12, 149-163.

Schaub, S. (2000) Teaching Java with Graphics in Cs1. *ACM SIGCSE Bulletin,* 32, 71-73.

Sedgewick, R. (2002) *Algorithms in Java*, Addison-Wesley.

Seppälä, O. (2004) Program State Visualization Tool for Teaching Cs1. *Third Program Visualization Workshop.* University of Warwick.

Shaffer, C. A., Cooper, M. & Edwards, S. H. (2007) Algorithm Visualization: A Report on the State of the Field. *Proceedinds of the 38th SIGCSE technical symposium on Computer science education* Covington, Kentucky, USA, ACM Press.

Shilling, J. J. & Stasko, J. T. (1992) *Using Animation to Design, Document and Trace Object-Oriented Systems*, Georgia Institute of Technology.

Simon, B., Anderson, R., Hoyer, C. & Su, J. (2004) Preliminary Experiences with a Tablet Pc Based System to Support Active Learning in Computer Science Courses. *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education.* New York, NY, USA, ACM.

Slator, B. M., Hill, C. & Del Val, D. (2004) Teaching Computer Science with Virtual Worlds. *Education, IEEE Transactions on,* 47, 269-275.

Smith, T. W. & Colby, S. A. (2007) Teaching for Deep Learning. Routledge.

Snyder, A. (1986) Encapsulation and Inheritance in Object-Oriented Programming Languages. ACM.

Stash, N. V., Cristea, A. I. & Bra, P. M. D. (2004) Authoring of Learning Styles in Adaptive Hypermedia: Problems and Solutions. *Proceedings of the 13th international World Wide Web conference on Alternate track papers \& posters* New York, NY, USA, ACM Press.

Stasko, J., Badre, A. & Lewis, C. (1993) Do Algorithm Animations Assist Learning?: An Empirical Study and Analysis. *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems.* Amsterdam, The Netherlands, ACM.

Stefik, A., Stefik, A., Fitz, K. & Alexander, R. (2006) Layered Program Auralization: Using Music to Increase Runtime Program Comprehension and Debugging Effectiveness

Layered Program Auralization: Using Music to Increase Runtime Program Comprehension and Debugging Effectiveness. IN Fitz, K. (Ed.) *Program Comprehension, 2006. ICPC 2006. 14th IEEE International Conference on.*

Stern, L. & Sterling, L. (1996) Teaching Ai Algorithms Using Animations Reinforced by Interactive Exercises. *the 2nd Australasian conference on Computer science education* The Univ. of Melbourne, Australia, ACM Press.

Steve, W., Patrick, H. & Myrtle, W. (1994) Filochat: Handwritten Notes Provide Access to Recorded Conversations. *Proceedings of the SIGCHI* *conference on Human factors in computing systems: celebrating interdependence.* Boston, Massachusetts, United States, ACM.

Stifelman, L. J. (1995) A Tool to Support Speech and Non-Speech Audio Feedback Generation in Audio Interfaces. *Proceedings of the 8th annual ACM symposium on User interface and software technology.* Pittsburgh, Pennsylvania, United States, ACM.

Stockley, D. (2003) E-Learning Definition and Explanation (Elearning, Online Training, Online Learning). http://derekstockley.com.au/.

Sweller, J. (2002) Visualisation and Instructional Design. *Proceedings of the International Workshop on Dynamic Visualizations and Learning.* Tübingen, Germany, Knowledge Media Research Center.

Sweller, J. (2008) Cognitive Bases of Human Creativity. *Educational Psychology Review,* 21, 11-19.

Tavangarian, D., Leypold, M., Nölting, K., Röser, M. & Voigt, D. (2004) Is E-Learning the Solution for Individual Learning? *Electronic Journal of e-Learning,* 2.

Triantafillou, E., Pomportsis, A. & Georgiadou, E. (2002) Aes-Cs: Adaptive Educational System Based on Cognitive Styles. *Adaptive Hypermedia and Adaptive Web-Based Systems.*

Tseng, J. C. R., Chu, H.-C., Hwang, G.-J. & Tsai, C.-C. (2008) Development of an Adaptive Learning System with Two Sources of Personalization Information. *Computers & Education,* 51, 776-786.

Tudoreanu, M. E. (2003) Designing Effective Program Visualization Tools for Reducing User's Cognitive Effort. *Proceedings of the 2003 ACM symposium on Software visualization.* San Diego, California, ACM.

Uskov, V. & Saad, A. (2000) Development of Online Undergraduate Object-Oriented Programming Curriculum. *Frontiers in Education Conference, 2000. FIE 2000. 30th Annual.*

Vasilyeva, E., Pechenizkiy, M., Gavrilova, T. & Puuronen, S. (2007) Personalization of Immediate Feedback to Learning Styles. *Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on.*

Vavoula, G. N. & Sharples, M. (2009) Lifelong Learning Organisers: Requirements for Tools for Supporting Episodic and Semantic Learning. *International Forum of Educational Technology & Society,* 12, 82-97.

Vickers, P. & Alty, J. L. (1996) Caitlin: A Musical Program Auralisation Tool to Assist Novice Programmers with Debugging. IN Frysinger, D. S. P. & Kramer, G. (Eds.) *ICAD: International Conferenceon Auditory Display.* Palo Alto, California, ICAD.

Vickers, P. & Alty, J. L. (2005) Musical Program Auralization: Empirical Studies. *ACM Transactions on Applied Perception (TAP),* 2, 477 - 489

Vickery, R. (2005) Teaching Object-Oriented Principles to 'a' Level Students. *Computer Education,* 5, 109-111.

Wang, K. H., Wang, T. H., Wang, W. L. & Huang, S. C. (2006) Learning Styles and Formative Assessment Strategy: Enhancing Student Achievement in Web-Based Learning. *Journal of Computer Assisted Learning,* 22, 207-217.

Warendorf, K. (1997) Adis-an Animated Data Structure Intelligent Tutoring System on the Www. *Information, Communications and Signal Processing, 1997. ICICS., Proceedings of 1997 International Conference on.*

Weidong, H., Peter, E. & Seok-Hee, H. (2009) Measuring Effectiveness of Graph Visualizations: A Cognitive Load Perspective. Palgrave Macmillan.

Wilson, B. G. (1995) Metaphors for Instruction: Why We Talk About Learning Environments. *Educational Technology,* 35, 25-30.

Wolf, C. (2003) Iweaver: Towards 'Learning Style'-Based E-Learning in Computer Science Education. *Australasian Computing Education Conference (ACE2003).* Adelaide, Australia, Australian Computer Society, Inc.

Woolf, B., Aïmeur, E., Nkambou, R., Lajoie, S., Parvez, S. & Blank, G. (2008) Individualizing Tutoring with Learning Style Based Feedback. *Intelligent Tutoring Systems.* Springer Berlin / Heidelberg.

Yerushalmi, E., Cohen, E., Heller, K., Heller, P. & Henderson, C. Instructors' Reasons for Choosing Problem Features in a Calculus-Based Introductory Physics Course. *Phys. Rev. ST Phys. Educ. Res.,* 6, 020108.

Ying, Z., Ying, Z. & Fernando, T. (2002) A Walk-through System for Building Acoustics Evaluation Based on Virtual Environment Technology

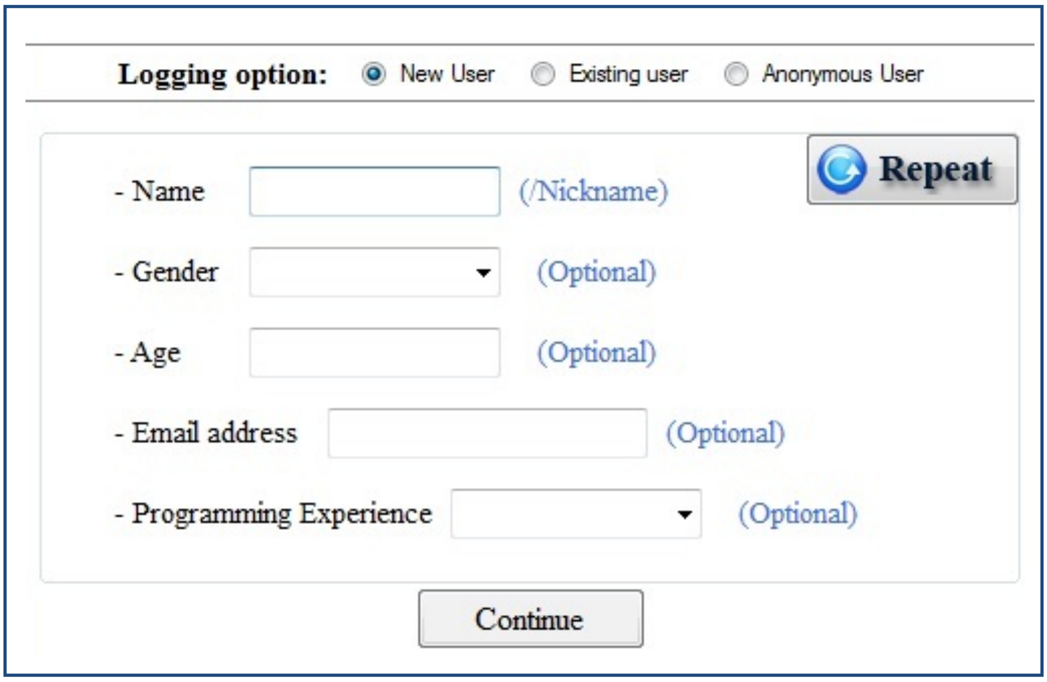
a Walk-through System for Building Acoustics Evaluation Based on Virtual Environment Technology. IN Fernando, T. (Ed.) *Industrial Technology, 2002. IEEE ICIT '02. 2002 IEEE International Conference on.*

Zualkernan, I. A., Allert, J. & Qadah, G. Z. (2006) Learning Styles of Computer Programming Students: A Middle Eastern and American Comparison. *Education, IEEE Transactions on,* 49, 443-450.

Zywno, M. S. (2003) Student Learning Styles, Web Use Patterns and Attitudes toward Hypermedia-Enhanced Instruction. *Frontiers in Education, 2003. FIE 2003. 33rd Annual.*

Appendices

Appendix A



Logging option: [✓ New User] Existing user Anonymous User

- Name		(/Nickname)
- Gender		(Optional)
- Age		(Optional)
- Email address		(Optional)
- Programming Experience		(Optional)

Appendix B

Technical Feedback

- "Value not found" error occurring even when the program finds the value and successfully appends it, when using "add after selected" function. Problem occurs as a result of user inserting nodes with letters and symbols instead of numbers (which works). Program cannot find the letters nodes.
- I am getting an error that the index is out of bound when I add random number for example: 12,22,11,24,32,2,1,10,35 then it is not proper binary tree as the program does not output it in the format of it. and it tells me that is out of bound even though I should be able to add more into it then only 1
- "A binary tree of depth greater than 3 (4 or more) places new nodes behind the traversal output boxes unless the user rearranges the tree manually.
- "Delete Selected" button seems to be non-functional
- I think it would have been really good to actually know which of them the tails are because some people still get confused between head and tail and think tail is ONLY the last value in the list. So it would be nice to have the objects somehow labelled to what they are
- The basic functionality of the tree construction is good (clearer to use than the other Binary tree tool). The fields at the bottom are useful also; though it would be better if they were overridden by new text as opposed to add at the end.
- "An option to sort the node values would be nice - such as ascending order from left to right or ascending order from top to bottom.

- Only notable problem found is with duplicate entries (i.e. two or more nodes with the same value).
- Could be made clearer how to make the node bigger and smaller by clicking on the top. Also can I not double click on the 'Node Name' words to make smaller? That could be easier
- Delete seems to work correctly on external nodes most of the time! Not working on an external node (89) without having attempted to delete an internal. Successfully deleted another external node (76)
- I manage to make 2 head nodes!
- If you take a snapshot, hit Save and then choose to Cancel it brings up an error message about ""Primary key duplicate entry
- It is good and helpful and it should clear the screen when the 3 buttons are pressed. Such as pre order traversal, in order traversal and post order
- Maybe blank out 'add left' and 'add right' when node hasn't yet been created.
- Found some bugs when trying to 'add left' or 'add right' more than once
- Maybe make it clearer you can double click elements to display more data
- A setAll button would be nice when adding data to Methods
- Objects can be hidden behind menus where they can’t be dragged
- Sometimes nodes overlap and so they aren't easy to see
- Delete button does not provide feedback if told to delete internal

- This one can get a bit messy as nodes are added, also I'm not sure how I did it, but I managed to create an exception when deleting a node and it completely crashed the program
- I managed to bug one node out somehow, it isn’t linked to anything. Also when I add something after a certain number of nodes, it doesn’t allow it.
- Trying to add a right-node to a node which already has one produces "Left node already exists."
- Very good tool.... though it gets a bit cluttered when a lot of objects are created and I can't find a way to delete them
- With 17 nodes present, attempting to add another threw an unhandled exception for index being out of array bounds
- The way in which nodes are added to the tree is not clear to me and so the tool doesn't seem that useful until you can construct a tree, exactly as you want it
- I would like to be able to edit/delete objects, for example being able to change the object name

Appendix C

Nontechnical Feedback

- Again, very useful to learn about traversing the trees
- Audio instructions and descriptions very useful, as is ability to add a node into an existing list
- Audio is good - it helps greatly with the task. Without it however it is hard to understand what is going on. Maybe it can be made easier with text as well?
- Excellent tool, especially for beginners. And very interactive. Code generation is amazing and the format is simple to understand.
- Good enough for first time use
- Good for making visualisations of linked lists very quickly. Graphics are very simple but very easy tool to use, although linked lists are reasonably easy to visualise
- Good tool for visualising and becoming more familiar with the concept of linked list
- Obvious and easy to use for making tree diagrams for exercise
- It helps me to visualise and understand the whole thing about binary trees and data structure. And integrated wiki also a good idea
- Much better than reading a pile of papers to understand things. And easy to use too.

- Reasonably simplified after a year's worth of programming with objects but a good idea in explaining objects at the beginning of the year when it was hard to visual them
- view as array and wiki great idea
- This part has provided me with a deep knowledge about linked list as I was finding it hard to acquire information from text books
- This section demonstrated me about tree traversal well. However, a video tutorial how to use would be heaven an "icing on the cake".
- This tool is very good for linked lists; the way they are drawn on screen is useful.
- Useful for visualizing the objects, though possibly a function for looking at classes and more difficult to grasp topics such as inheritance, etc would me more useful than understanding simple objects
- Useful Java beginners' tool. Good data type checking, although Strings are not checked for "".
- Very easy for making a quick binary tree
- Very good for getting to know how the binary trees work. Especially the comparison is done automatically so when people don't know where the inserted node should appear in the tree they can use this to get the answer which is very helpful.

Appendix D

	The learning tool interface was easy to use		
Q1	1 Strongly Agree	**37.50%**	6
	2 Agree	43.80%	7
	3 Somewhat Agree	12.50%	2
	4 Somewhat Disagree	6.30%	1
	5 Disagree	0.00%	0
	6 Strongly Disagree	0.00%	0

	The audio instructions were clear and easy to understand		
Q2	1 Strongly Agree	**25.00%**	4
	2 Agree	37.50%	6
	3 Somewhat Agree	31.30%	5
	4 Somewhat Disagree	6.30%	1
	5 Disagree	0.00%	0
	6 Strongly Disagree	0.00%	0

	The textual instructions were easy to understand		
Q3	1 Strongly Agree	**18.80%**	3
	2 Agree	25.00%	4
	3 Somewhat Agree	25.00%	4
	4 Somewhat Disagree	25.00%	4
	5 Disagree	6.30%	1
	6 Strongly Disagree	0.00%	0

	The provided visualisations covered the main data structures that considered important in this module		
Q4	1 Strongly Agree	**12.50%**	2
	2 Agree	50.00%	8
	3 Somewhat Agree	31.30%	5
	4 Somewhat Disagree	0.00%	0
	5 Disagree	6.30%	1
	6 Strongly Disagree	0.00%	0

	There was enough information about how to create and interact with object visualisation		
Q6	1 Strongly Agree	**13.30%**	2
	2 Agree	40.00%	6
	3 Somewhat Agree	33.30%	5
	4 Somewhat Disagree	13.30%	2
	5 Disagree	0.00%	0
	6 Strongly Disagree	0.00%	0

	There was enough information about how to create and interact with Linked List visualisation		
Q7	1 Strongly Agree	**25.00%**	4
	2 Agree	25.00%	4
	3 Somewhat Agree	25.00%	4
	4 Somewhat Disagree	25.00%	4
	5 Disagree	0.00%	0
	6 Strongly Disagree	0.00%	0

	There was enough information about how to create and interact with Binary Search Tree visualisation		
Q8	1 Strongly Agree	**25.00%**	4
	2 Agree	37.50%	6
	3 Somewhat Agree	12.50%	2
	4 Somewhat Disagree	25.00%	4
	5 Disagree	0.00%	0
	6 Strongly Disagree	0.00%	0

	The Binary Tree Traversal task was useful and important to test and enhance your knowledge about tree traversal		
Q9	1 Strongly Agree	**43.80%**	7
	2 Agree	25.00%	4
	3 Somewhat Agree	31.30%	5
	4 Somewhat Disagree	0.00%	0
	5 Disagree	0.00%	0
	6 Strongly Disagree	0.00%	0

The visualisations helped in creating mental model of proposed data structures			
Q5	1 Strongly Agree	**37.50%**	6
	2 Agree	31.30%	5
	3 Somewhat Agree	25.00%	4
	4 Somewhat Disagree	6.30%	1
	5 Disagree	0.00%	0
	6 Strongly Disagree	0.00%	0