THE UNIVERSITY
of ADELAIDE

School of Computer Science

Introduction to Programming in Alice

Unit and Lesson Plans

Prepared by:
Dr. David Hemer

January 20, 2012

Unit Overview

Unit Plan Title Introduction to Programming

Year Level Year 11

Duration 5 weeks, 2 double lessons per week + 3 weeks for group project

Unit Questions

The questions addressed by this unit of work are:

- How can a complex problem be solved by breaking it down into a sequence of simpler steps?
- How can problems with choice be solved using conditionals?
- How can repeated actions be performed using loops?
- How can real world entities be represented using classes and objects?
- How can algorithmic constructs be represented in an application program?

Curriculum-framing Questions

The Key Questions from the SACE Stage 1 IT curriculum [19] addressed for this unit of work are as follows:

1. How is an application program designed?
2. How is an application program developed?
3. How is the design of an application program tested and how are errors resolved, if necessary?

A more detailed analysis of the SACE Stage 1 IT curriculum key questions and concepts is given in Appendix A.

Content Questions

The content questions addressed by this unit of work are:

- How are 3D worlds created in Alice and how are objects added to these 3D worlds?
- How are objects controlled in Alice using an individual action?
- How can complex behaviour in the Alice world, including interaction between objects, be represented in pseudo-code as a sequence of actions and/or concurrent actions?
- How can these designs be programmed in Alice?
- How can the behaviour of objects be conditionally controlled?
- How can actions be repeated either a fixed number of times or until some condition is satisfied?

- How can a complex problem be decomposed into simpler parts?
- How can user interaction (such as mouse button and keyboard presses) with the objects be handled?

Unit Summary

This document provides a unit plan for a component of work on Application Programming meeting part of the requirements of the SACE stage 1 Information Technology curriculum [19]. The broad aims of this unit of work is to introduce students to the main fundamental concepts of programming languages, such as variables, basic data structures, sequence, selection and iteration.

In order to engage students and to maintain an ongoing interest in computer programming, these concepts will be taught and learned using the Alice 3D programming environment [6]. Alice allows students to create 3D worlds containing a variety of objects that can be programmed to interact with each other in a variety of ways. Alice enables students to learn the aforementioned programming language concepts without students having to be burdened initially with quirks of program language syntax. Alice has the benefit that it introduces students to object oriented design and programming early, but it does so in an unobtrusive manner.

Teaching and Learning

Learning outcomes

On completion of this unit of work, students should be able to:

- explain the concepts of variables, sequence, selection, iteration and objects
- create a virtual 3D world with a variety of objects using the Alice programming environment
- apply these programming language constructs on their own, or in simple combinations to program the behaviour of objects in their virtual world

Lesson Outline

Table 1 shows a lesson outline for the Application Programming unit. It is assumed that each lesson is a double lesson (approximately 1.5 hours in duration). In the case where single lessons are used, the outline will need to be refined appropriately.

Lesson	Learning Objectives
Lesson 1	**Introduction to Alice**
Lesson 2	**Sequential, Parallel and Simple Loops**
Lesson 3	**Variables and Functions**
Lesson 4	**Methods and Parameters**
Lesson 5	**Advanced Functions**
Lesson 6	**Practical Test 1**
Lesson 7	**Conditional Statements**
Lesson 8	**Indefinite Loops**
Lesson 9	**Events and Event Handling**
Lesson 10	**Practical Test 2**
Lessons 11–16	**Group Project**
End of Semester	**Practical Exam**

Table 1: Lesson outline for Application Programming unit

Strategies

A variety of teaching and learning strategies will be employed throughout the unit. We describe each of these strategies below, together with an indication of the technologies employed.

Instructional teaching: in each lesson now programming concepts will be introduced by the teacher before the students practice using these concepts in a small group setting. To demonstrate the concepts the teacher can use a simple running that they will add to a refine as the unit proceeds.

A live demonstration will be used to introduce new concepts. The teacher will require a teacher PC/laptop connected to a projector to conduct the demonstrations. Alternatively, if available in the computing lab, the teacher can use a electronic whiteboard (e.g. Smartboard) to run the demonstrations. Another possibility is to use a tool such as LANSchool [1] to display the teacher's demonstration on the student's PC's in the computing lab. LANSchool has the added benefit that it allows the teacher to easily demonstrate the work of other students to the rest of the class.

Cooperative learning: this unit will have a large emphasis on small group work. Ideally students will work in pairs, however depending on the size of the class and the available resources, this may be extended to working in groups of three.

To ensure all students contribute to the work of their group, the teacher should ensure that roles are well defined — for example one student may be responsible for writing down the design, while the other student is responsible for coding the design. Both students would be responsible for testing and debugging the code. The teacher must also ensure that students take it in turn to take on a particular role. Furthermore, to ensure that all students in a group have an understanding of the concepts, the teacher must ensure that all students, are in turn asked to explain their design/code etc.

Guided discovery activities: after the teacher has demonstrated a particular concept, student's will be asked to design and code a solution for a related problem. Whilst this will be based on the demonstrated concept, it may expand on what was demonstrated, or may require the concept(s) to be applied in a different context. Students may also be required to use experiment with and use concepts that have not yet been explained in class.

Blended learning: students will be required to report on their progress for their group project throughout the duration of the unit. To facilitate this, we propose using online technologies such as wikis, blogs and discussion forums.

Blogs will be used to report on the progress of the group throughout the project — this reporting will include mandatory milestone reports (at predefined times), together with a final report. Similarly, a blog could be used by individuals to record their personal reflections during and at completion of the project.

A Wiki will be used as a way for students to share their knowledge and understanding of the Alice programming environment. Each student in the class will be required to contribute to the Wiki (see Assessment for more details).

A discussion forum will be used by the students to ask questions about certain aspects of the group project and the Alice programming environment.

Activities

Each lesson will include a small number of group-based tasks designed to achieve the learning objectives for the lesson. To accommodate different students of differing abilities, a variety of tasks will be available in each lesson of varying levels of difficulty, including "challenge problems". Challenge problems will might involve the use of more advanced features of the language, or may involve using the language constructs in an atypical way.

Each lesson will also include time for students to apply the concepts learned in class to their group project.

The activities done throughout the unit will develop a range of skills, including:

Design: Students write down a design for their problem using a simple pseudo code similar to that used in the Alice textbook [8]. Students will be expected to complete a design for their problem and to evaluate this design before they do any coding for the problem.

Coding: Once the students have completed their design they will then write code that satisfies that design. Students will be encouraged to use an incremental and evolutionary approach for more complex problems in which they code, test and debug part of the problem before moving on to the next part.

Testing and Debugging: Students will be required to test that their program performs its intended behaviour. This testing process will be done by running the program and observing its behaviour. Where the desired behaviour is not met, students will need to determine why the behaviour has not been met and how to correct the program.

Documenting: Students will be required to document processes they have used and the products they have produced. This documentation will include the group project milestones, individual reflective reports and class wiki.

Presenting: In the last week of the unit, students will be required to present their group project. The presentation should include a demonstration of the program (does not need to be complete — instead they are demoing a work-in-progress), as well as a description of the process they have used in developing the program.

Reflecting: Students will be required to reflect on what they have learned during the unit. Their reflections should include:

- What they have found most difficult about application programming in Alice.
- What aspect of application programming in Alice did they find easiest to learn.
- What are the benefits of working in a group when developing a program.
- What challenges did you face when doing group work during the project.

Prerequisite skills

This unit of work assumes basic computer literacy. Before commencing this unit of work, students should:

- Be familiar with the operating system used in the computing lab.
- Be able to create and manage folders for storing Alice programs, using networked storage.
- Be able to use flash drives to transfer and backup their work.
- Be familiar with the LMS tool(s) (e.g. Moodle, Edmodo etc) used within the unit.

Professional Learning

Teachers should are expected to have the following knowledge and skills prior to teaching this unit:

- In depth understanding of programming language concepts, including object orientation.

- Working knowledge of Alice 3D programming environment.
- Basic system administration skills to ensure that students are saving and backing up their work appropriately.

Materials and resources

Technology - Hardware

⋆ Computer Suite, with at least 1 computer for every two students
⋆ Networked storage, with regular backup regime.
⋆ Data projector
⋆ Projector screen/electronic whiteboard
⋆ Server for hosting Moodle software

Technology - Software

⋆ Windows or Mac OS (see below for more details)
⋆ Alice 3D programming environment (see Fig. 1). The Alice programming environment runs on Windows or Mac operating systems. The Alice website lists the following system requirements:
  ◇ Windows 7, Vista, XP, or 2000 — Mac OS X 10.4+
  ◇ Intel Pentium II or equivalent processor — PowerPC or Intel processor
  ◇ A VGA graphics card capable of high (16 bit) color and 1024x768 resolution (3D video card recommended)
  ◇ 512MB of RAM (1GB recommended)
  ◇ A sound card
⋆ Web Browser (preferably Firefox, Safari or Chrome)
⋆ Integrated LMS with discussion forum, wiki and blogging capabilities, such as Moodle software [2]. Alternatively individual components such as Edmodo [9] (secure social networking for discussion) and elgg [7] (for blogging).
⋆ Quicktime player for viewing videos generated by Alice software.

Textbooks

See `http://www.alice.org/index.php?page=documentation` for a more complete listing:

- Learning to Program with Alice, Wanda Dann, Stephen Cooper, and Randy Pausch, Prentice Hall. [8]
- Fluency with Alice: Workbook for Fluency with Information Technology: Skills, Concepts, and Capabilities by Robert Seidman, Phil Funk, Jim Isaak, Lundy Lewis
- An Introduction to Programming Using Alice by Charles W. Herbert

Other Resources

- Alice Community Forums `http://alice.org/community/`
- Alice Ancestor Project YouTube channel `http://www.youtube.com/user/ancestorproject`
- Programming in Alice YouTube channel `http://www.youtube.com/user/drdave785`
- Professor Zelda's YouTube channel `http://www.youtube.com/user/ProfessorZelda`
- SFunk's YouTube channel `http://www.youtube.com/user/sfunk1992`

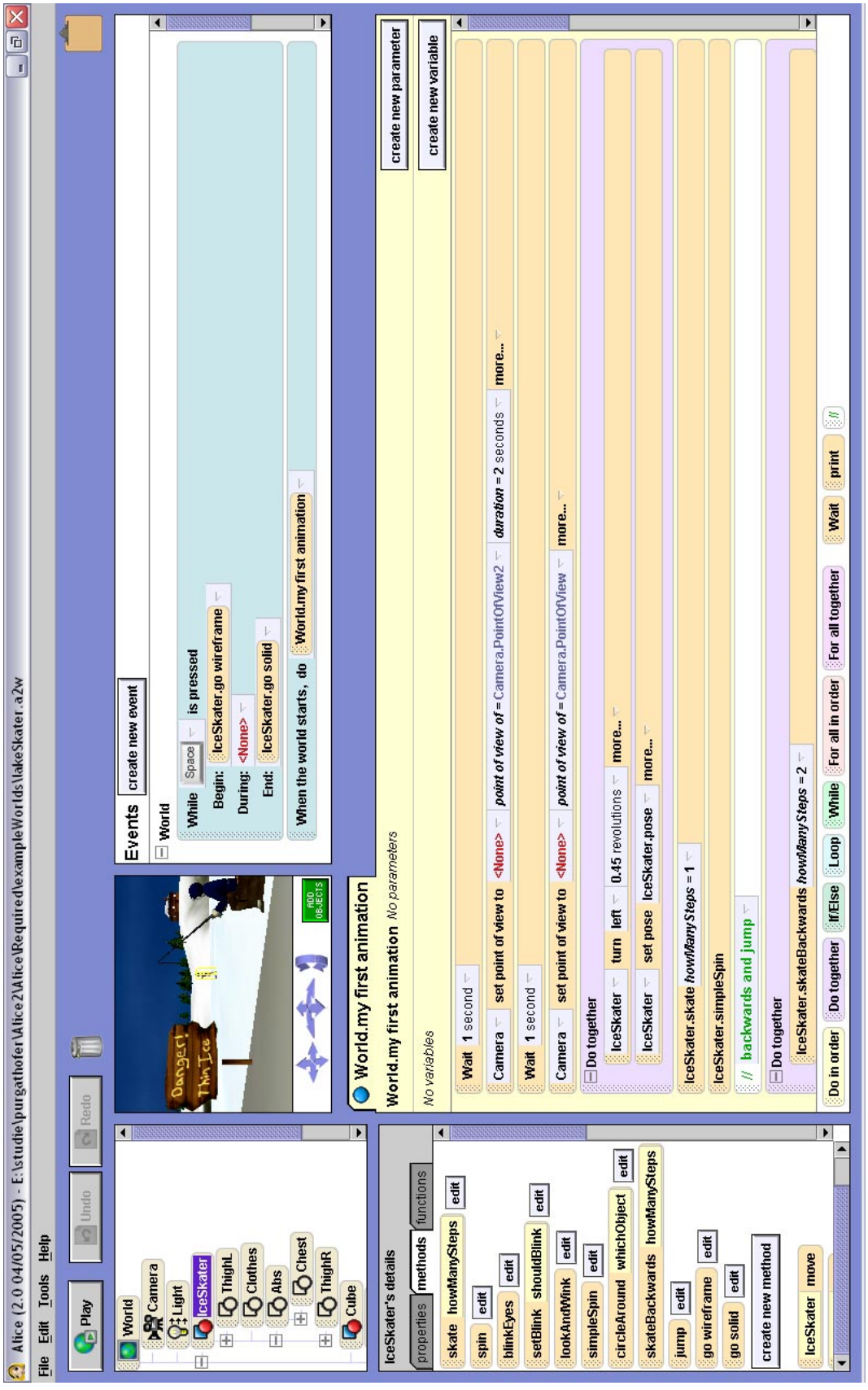

Figure 1: Screenshot of Alice 3D programming environment

Rationale for selection and use of ICTs

Choosing a programming environment

The key aims of this unit are to introduce students to applications programming – clearly we need some for of ICT to do this, i.e. programming environment.

The key concern is *what* programming language/environment to choose to best facilitate learning and engagement. The rationale for choosing a programming environment should include:

- ◇ Ease of use and appropriateness for year level, including learning curve
- ◇ Portability and availability
- ◇ Coverage of key concepts, including sequential constructs, conditionals, loops and objects (for objects early learning approach)
- ◇ Documentation and supporting material
- ◇ Engagement

Programming languages such as C++ and Java cover the key learning concepts, and are generally available across a variety of platforms. However both have a steep learning curve and students can easily get bogged down with syntactic issues, whereas the focus should be on the semantics of programming languages. Furthermore at an introductory level it is difficult to get beyond very simple toy examples, thus it will be far more challenging to engage the students.

Javascript is slightly more forgiving in terms of syntax and the mechanics of interacting with the user are easier to learn, so in comparison to C++ and Java students could potentially progress towards the development of more interesting and engaging applications more rapidly. However to develop anything interesting students would also require a detailed knowledge of HTML.

A number of programming environments have been developed with the aim of easing students into programming. The strengths of Alice as a programming environment include:

- ⋆ Portable integrated development environment;
- ⋆ Drag and drop construction of programs removes the burden of syntax checking enabling students to focus on semantics;
- ⋆ Explicit focus on objects as real world entities – provides a gentle introduction to object orientation;
- ⋆ Students can creatively build applications.

A study by Moskal [17] looking at at-risk computer science students shows a significant improvement in attitude (confidence in programming, liking of programming and sense of creativity) towards programming from students who use the Alice programming environment in comparison to two separate control groups who did not use Alice.

Quantitative [4] and qualitative studies [14] show the benefits of conducting programming in Alice in pairs, versus individual programming. Wang [22] shows small improvement in test performance in Alice group versus a control.

Wikis and blogs

Godwin-jones [12] describes the use of blogs in classrooms as online journal, which also allows for interaction with other members of the class who can read and comment on blog entries. This thus enables students to share knowledge and experiences. Blogs can be used to capture the evolving design and thought processes that occur during the development of the group project. The evolving nature of this information gives more insight into the design process than a static design document at the end of the project. The fact that it is easily accessible to the teacher and other students means that the whole class can provide useful feedback.

Wikis have a number of educational uses [5, 10, 13, 18]. In the context of this unit, a wiki can be used to shared knowledge of the Alice programming environment with the entire class. The benefit of this, is that as one student discover a useful feature of the language they are able to share it with the rest of the class. By encouraging students to contribute to the wiki, it is far more likely that this newly discovered knowledge is shared amongst the entire class. This can be done by other means (e.g. giving a class presentation), however the use of a wiki means the sharing is more immediate and more accessible. By reviewing the contents, the teacher is also able to ensure the information on the wiki is correct. This helps reduce the potential of students sharing misinformation amongst the class via ad-hoc verbal means.

Differential Learning

Girls in IT

Girls are grossly under represented in the fields of IT, Computer Science and Software Engineering although there is evidence to suggest that positive experiences with IT can lead to positive attitudes [3]. We conjecture that engendering positive attitudes can assist in addressing the gender inbalance in IT. Research suggests that the use of the Alice programming environment, where students can engage in story telling and scenario development assists in the engagement and retention of girls in computer programming [15].

Learning Styles

VAK/VARK [11] tests have become increasingly popular in determining whether a student's preferred learning style/mode is visual, auditory, reading/writing or kineasthetic. To test preferred learning styles VAK/VARK questionnaires are often used in high schools. The credibility of such questionnaires has been questioned in the literature [21]. However it would seem beneficial to support multiple learning modes.

Table 2 lists some of the different activities that could engage the different learning modes. The list is not intended to be comprehensive, but it should demonstrate that it is possible to support all four modes. Alice provides good support for visual learning through the manipulation of objects in a 3D world. This is a big advantage over traditional programming languages where much of the focus is on the syntax of a language, which can be argued provides a better fit for the reading/writing mode. Programming is typically a desk-bound activity so opportunities for kineasthetic learning are limited (arguing that moving a mouse around is kineasthetic is probably a long stretch). However Alice again presents opportunities – is develop movement and interaction of objects, especially human-like characters, students can act scenarios and check movement using their own body parts.

A variety of other categorisations of learning styles have been proposed in the literature. One example is the Learning Style Inventory (LSI) of Kolb [16]. Kolb proposes four main learning styles: diverging, assimilating, converging and accommodating. Diverging students are best at viewing concrete situations from many different points of view, and are well suited to brainstorming activities. They are interested in people, prefer to work in groups. Assimilating students are interested in ideas and abstract concepts and are not so interested in the practical value of these ideas. Converging students are best at finding practical uses for ideas and theories and prefer to deal with technical tasks. Accommodating students learn from hands-on experience and prefer to work with other people to get assignments done.

Special Needs

Within the class we will inevitably have students at different levels of ability. To cater for the *advanced* students (including *gifted* students), we would ensure that there are extension exercises available in each lesson. These exercises will still focus on the main concepts taught in the lesson, however they would involve more complex scenarios and may require the student to do additional reading or research to learn about advanced language features. A portion of each lesson will also be spent on the group project – advanced students may have more time to work on the project, the expectation would be that such students be able to develop more complex scenarios. The rest of the class is not left behind since they are still learning the key programming concepts. Advanced students will also be able to share some of this advanced knowledge with the remainder of the class by contributing to the class wiki. However, we are

Learning Mode	Activities
Visual	Demonstration of Alice programs (videos)
	Visual testing of Alice programs
Auditory	Teacher instruction introducing new concepts
	Student presentation of scenarios and designs
Reading & Writing	Writing scenario designs
	Development and use of wiki help pages
	Writing personal reflections in blog
Kineasthetic	Acting out scenarios
	Demonstration and development of code on electronic white-board
	Testing of movement of Alice objects by observing our own movement

Table 2: VARK activities in the programming class

careful to ensure that the whole class contributes to the wiki – this is not just an exercise for the advanced students.

For students with Asperger's syndrome, group-work can present a daunting challenge. Safran [20] recommends that teachers "pay careful attention to the makeup, structure, and process of groups". She further notes that teachers should avoid self-selection of groups and to make sure students are grouped appropriately, in particular avoiding grouping bullies or aggressive students with students with Asperger's, instead grouping them with understanding students. Following is part of a speech by a student with Asperger's [20]:

> Don't let kids pick their own groups for group-work . One of the important things group-work is supposed to teach is how we can work with diverse people, who we don't get along well with. Allowing kids to pick their own groups defeats the purpose of this. Certain kids are always left out and isolated. It's really embarrassing when no one wants you and you either have to work alone or the teacher has to find you a group.

To support this, the teacher will need a good knowledge of their students before groups are assigned. This may be problematic if this unit is at the beginning of the course. If this is the case the teacher may need to defer allocating the groups until the second week the unit, whilst conducting temporary group activities in the first week for observation purposes.

Assessment

Formative assessment will be ongoing throughout the unit. During each class students will be required to explain their current work to the teacher to ensure that they have an understanding of the concepts being taught. Assessment will involve ticking off tasks as they are satisfactorily completed by the groups. Students will not only be assessed on their ability to write code, but also on other aspects, such as design and testing.

Summative assessment will also be conducted throughout the unit. A summary of the summative assessment is given below. A detailed description of the group project assignment is given with the lesson resources.

1. Practical Tests x 2 [25% each]
2. Group project [50%]

In addition teachers may also include a practical examination as part of their end of semester examination.

Lesson Plans

Lesson: Introduction to Alice

Subject Year 11 Information Technology	**Unit** Introduction to programming
Topic Introduction to Programming	**Lesson Duration/Date** 2 hour double lesson
Lesson Questions ⋆ What is software and how is it developed? ⋆ What is computer programming? ⋆ What is a computer programming language and how have they evolved over the years?	**Objectives** ⋆ Students can list a variety of applications and domains that use software ⋆ Students can explain in high-level terms how software is developed ⋆ Students can create a new world in Alice ⋆ Students can create and position objects in Alice ⋆ Students can program individual actions in Alice and/or modify object attributes
Introduction (20 minutes) In the lesson opener the concepts of software, computer programming and programming languages will be briefly introduced. The teacher should then briefly introduce the Alice programming environment, ensuring that all students have access to Alice. This introduction should be kept short, with students given the opportunity to explore Alice for themselves. Teachers may wish to show one of the introductory Alice videos available on the Alice website `http://www.alice.org/index.php?page=what_is_alice/what_is_alice`.	
Main Activity Task (70 minutes) The main task for this lesson will be an introduction to the Alice programming environment. An activity is described in Alice Tutorial: Introduction to Alice. However students should be encouraged to work through the built-in tutorials prior to attempting the workshop (or they may indeed work through these tutorials instead of the workshop).	

Resources and Materials	**Differentiation**
⋆ Alice textbook ⋆ Computer suite with Alice 2.2 software ⋆ Desktop/laptop connected to data projector ⋆ Alice Tutorial: Introduction to Alice	An extension exercise is included on the worksheet. In general students should be given some freedom to explore Alice in this lesson. However more direction is required the teacher may develop additional exercises, or point students towards some of the more advanced features (e.g. working with the camera; setting opacity and isShowing properties; vehicles; light etc.).
Lesson Closer (10 minutes) In the lesson close, the teacher should get the student's initial impressions of the Alice environment. Students should be given an opportunity to ask questions and comment on Alice. This will give the teacher the opportunity to clear up any initial misconceptions about the Alice environment.	
Assessment	**Lesson Comments**
In order to determine previous programming experience the teacher shoud conduct a short survey of students during the first lesson. This could be conducted using the school's learning management system (e.g. Moodle or Edmodo). Results from this may be used to adapt the pace of the lessons in this unit.	This lesson should provide students with a fair degree of freedom to explore Alice. Teachers may wish to spend more time describing the history and significance of computer programming. If time permits an additional lesson may be conducted prior to this one. This lesson should motivate the importance of computer programming, then give students an opportunity to explore some aspect of computer programming.

Lesson Plan: Sequential, Parallel and Simple Loops

Subject	**Unit**
Year 11 Information Technology	Introduction to programming
Topic	**Lesson Duration/Date**
Sequential, Parallel and Simple Loop	2 hour double lesson
Lesson Questions	**Objectives**
⋆ How can simple programming constructs be combined to develop more complex programs? ⋆ What does it mean when programming constructs are implemented sequentially? ⋆ What does it mean when programming constructs are implemented in parallel (or concurrently)? ⋆ How can programming commands be repeated a fixed number of times?	⋆ Students can design a simple scenario using a combination of sequential and parallel actions ⋆ Students can implement their design in Alice using an appropriate combination of constructs. ⋆ Students can use simple loops to repeat an action multiple times.

Introduction (20 minutes)

In the lesson opener the concepts of sequential and parallel programming constructs will be introduced.
To demonstrate the concepts, the teacher can use the simple example of a bouncing beach ball.

⋆ Firstly demonstrate sequential combination to show the beach ball moving up *then* moving down. Show a simple design, then demonstrate by programming in Alice.
⋆ Next discuss how the design and code may modified to get the ball to move forward whilst moving up and down. Initially only use the sequential construct. Then demonstrate using the parallel construct to get the ball to move forward at the same time as it moves up and down. Be careful to highlight the importance of getting the timing correct.
⋆ Finally demonstrate the use of a simple loop to get the ball to bounce multiple times.

Main Activity Task (70 minutes)

The main activity for this lesson will be two design and coding tasks as described in Alice Tutorial: Sequential, parallel and simple loop constructs.
The first activity focusses on sequential and parallel constructs. Students should begin with a simple partial solution that uses the do in order construct. Once they have this working they can extend their code using the do together construct. Students should be encourage to experiment with different durations for the individual commands.
In the second task, students will use do in order, do together and simple loop constructs. Depending on the length of the lesson and the progress made by students, the teacher can extend this task by getting the students to add additional animations or extra interaction between the two objects.
Students can either work alone on this task or work in pairs. If students work in pairs then the teacher must ensure that they are both taking an active role. One way of ensuring that roles are shared is for students to swap the driver and navigator roles after the first task is complete.

Resources and Materials

- ⋆ Alice textbook
- ⋆ Computer suite with Alice 2.2 software
- ⋆ Desktop/laptop connected to data projector
- ⋆ Alice Tutorial: sequential, parallel and simple loop constructs.
- ⋆ Video tutorial: basic programming constructs `http://www.youtube.com/watch?v=R-2wo-9Sb3g`

Differentiation

An extension exercise is included on the worksheet. In this exercise students will use the same constructs used in the main two tasks. However the description of this task is less detailed, thus requiring students to think more about the design and how they can break the problem into smaller steps.

Lesson Closer (10 minutes)

In the lesson close, solutions to the two main tasks will be demonstrated. Students should be encouraged to share their solutions. If LanSchool or similar software is available, the student's solution can be projected to the rest of the class. Difficulties and challenges faced by the students should also be discussed at this stage. Students should also be given an opportunity to share any useful features that they have discovered in Alice.

Assessment

As a means of formative assessment the teacher should check on the progress of all students, marking off tasks as they are completed. For these tasks a simple checklist should suffice. Additional comments should also be recorded.

Lesson Comments

Given that this lesson is still early in the unit, students may be tempted to explore and play with Alice. However it is important that students complete the first two tasks on the worksheet, hence checking the progress of all students to ensure that they remain on task is essential. Having said that, students who do complete the main tasks should be encouraged to explore the Alice development tool in more detail.

Lesson Plan: Variables and Functions

Subject	**Unit**
Year 11 Information Technology	Introduction to programming
Topic	**Lesson Duration/Date**
Variables and Functions	2 hour double lesson
Lesson Questions	**Objectives**
⋆ What is a variable in a program? ⋆ How can variables be used to store data in our program? ⋆ How can variables be used in functions and methods? ⋆ What is the difference between local and global variables? ⋆ What are functions and how can they be used in our program? ⋆ How can functions be used to get input from the user?	⋆ Students can explain why variables are necessary in computer programs ⋆ Students can design and implement programs that use local and global variables to store data ⋆ Students can use input functions to get information from the user ⋆ Students can use variables in function and method calls

Introduction (20 minutes)

In the lesson opener the teacher should explain the concepts of functions and variables. For functions the teacher should emphasise that fact that Alice supports pure functions with no side effects, however other languages allow side effects in their functions.

To demonstrate the concepts, the teacher can use the user input example (see `http://www.youtube.com/watch?v=gOU0TsVP_yM`).

⋆ This example demonstrates the use of local variables, together with user input functions and string utility functions.
⋆ Teachers may initially only demonstrate the first question from Bob, leaving the number input and conversion for in the later or another lesson.
⋆ Teachers may additionally demonstrate global variables by creating variables in the `World` object.

The teacher may wish to demonstrate a simpler use of variables before looking at the user input example.

Main Activity Task (70 minutes)

The main activity for this lesson will be two design and coding tasks as described in Alice Tutorial: Variables and Functions. In the first task, students will use local variables, as well as user input and string utility functions. Students may need additional guidance in how to use the string utility functions.
In the second task, students will use the random number function to simulate dice rolls. Step 3 involves converting a number to a string (showing only the whole number part) – the teacher may wish to demonstrate this to students prior to them working on the task. Alternatively they can be referred to the user input video tutorial `http://www.youtube.com/watch?v=gOU0TsVP_yM`. The second half of the task requires students to use a local variable to accumulate a running total of the dice rolls. The teacher should ensure that the random number function is used correctly – omitting the integerOnly option is the most common cause of problems.

Resources and Materials

- Alice textbook
- Computer suite with Alice 2.2 software
- Desktop/laptop connected to data projector
- Alice Tutorial: Variables and Functions.
- Video tutorial: basic programming constructs `http://www.youtube.com/watch?v=R-2wo-9Sb3g`

Differentiation

An extension exercise is included on the worksheet. In this exercise students will use string utility functions, together with time functions to display the current time.

Lesson Closer (10 minutes)

In the lesson close, solutions to the two main tasks will be demonstrated. Students should be encouraged to share their solutions. If LanSchool or similar software is available, the student's solution can be projected to the rest of the class. Difficulties and challenges faced by the students should also be discussed at this stage. Students should also be given an opportunity to share any useful features that they have discovered in Alice.

Assessment

As a means of formative assessment the teacher should check on the progress of all students, marking off tasks as they are completed. For these tasks a simple checklist should suffice. Additional comments should also be recorded.

Lesson Comments

It is important that students become familiar with the various world level functions provided in Alice. Use of the string utility functions in Alice is a bit clumsy, but students should be encouraged to use these functions correctly in order to get correctly formatted output, in particular ensuring that spaces are used where appropriate and that numbers are converted correctly.

Lesson Plan: Methods and Parameters

Subject	**Unit**
Year 11 Information Technology	Introduction to programming
Topic	**Lesson Duration/Date**
Methods and Parameters	2 hour double lesson
Lesson Questions	**Objectives**
⋆ What is a method in an object oriented program? ⋆ How can methods be used to break a problem into manageable parts? ⋆ What is a parameter? ⋆ How can parameters be used to develop methods that can be used in different ways?	⋆ Students can define new methods for existing classes ⋆ Students can define methods with parameters ⋆ Students can use these methods in their programs

Introduction (20 minutes)

In the lesson opener the teacher should explain the concepts of methods and parameters. The teacher should firstly review the existing built-in methods that each object has, then demonstrate some of the additional methods that select objects include. The teacher may take this opportunity to demonstrate the heBuilder and sheBuilder objects that include a number of built in animation methods (see `http://www.youtube.com/watch?v=ulGpD7DEQns`). The teacher should demonstrate methods that do not have any parameters, together with those that do.
The teacher should then explain that programmers can also define their own methods for any of the objects. Before demonstrating how this is done, the teacher should explain why this is useful, in particular emphasising that using methods to decompose a program into smaller parts is a critical aspect of program design.
The teacher should then demonstrate how new methods can be defined, firstly without parameters and then with parameters.

Main Activity Task (70 minutes)

The main activity for this lesson is two design and coding tasks as described in Alice Tutorial: Creating Methods. In the first task students will create a object-level method that uses the built-in object methods. Students will begin by creating a method without parameters and then modify the method to introduce two parameters.
In the second task students will create methods for driving and steering a Humvee vehicle, these methods should then be called within the main method to move the Humvee around a building.

Resources and Materials	**Differentiation**
⋆ Alice textbook ⋆ Computer suite with Alice 2.2 software ⋆ Desktop/laptop connected to data projector ⋆ Alice Tutorial: creating methods	As an extension exercise to the worksheet students select an object of their choice, identify 4–5 suitable methods and provide implementations for these methods.

Lesson Closer (10 minutes)

In the lesson close, solutions to the two main tasks will be demonstrated. Students should be encouraged to share their solutions. If LanSchool or similar software is available, the student's solution can be projected to the rest of the class. Difficulties and challenges faced by the students should also be discussed at this stage. Students should also be given an opportunity to share any useful features that they have discovered in Alice.

Assessment	**Lesson Comments**
As a means of formative assessment the teacher should check on the progress of all students, marking off tasks as they are completed. For these tasks a simple checklist should suffice. Additional comments should also be recorded.	Confusion between local variables and parameters often causes students problems. Teachers should be very clear in making the distinction between these two language constructs, ensuring that students use the constructs correctly. The activities in this lesson do not place much emphasis on design, with the main tasks dictating what methods should be developed. Teachers may wish to include additional exercises that are more open ended allowing more emphasis on the design aspects of programming. This may indeed be included as an additional lesson (time permitting) or could be explicitly included as a task for the final project.

Lesson Plan: Advanced functions

Subject	**Unit**
Year 11 Information Technology	Introduction to programming
Topic	**Lesson Duration/Date**
Object Variables and Functions	2 hour double lesson
Lesson Questions	**Objectives**
⋆ What is an object variable? ⋆ What is a global variable? ⋆ What is the difference between local, object and global variables? ⋆ How can object-level functions be used to measure size and distance?	⋆ Students can design and implement programs using object variables and global variables ⋆ Student can use object-level functions

Introduction (20 minutes)

In the lesson opener the teacher should review local variables and functions. The teacher should then discuss (informally or via an example) the concept of variable scope, noting that a local variable can only used within the method or function that it is defined in. The teacher should then demonstrate a simple example of a program that uses an object variable. This should then be contrasted with the use of global variables (represented in Alice by variables in the world object).
Next the teacher should list and describe the various object-level functions available in Alice. Object-level functions could demonstrated using a simple collision detection example (see for example `https://www.youtube.com/watch?v=ORkkg9ZVdIA`. This video demonstrates the use of the **distance to** function to accurately determine the distance between two objects.

Main Activity Task (70 minutes)

The main activity for this lesson consists of three design and implementation tasks as described in Alice Tutorial: Object Functions and More Variables.
In the first task, students will use object variables to keep track of the distance travelled by a car. Step 3 of the exercise could be omitted, or left as a extension exercise or homework to ensure that students have sufficient time to work on the other two exercises in class.
The second task continues on from the first task. In this task students use a global variable (world variable) to keep track of the total distance travelled by two vehicles. Because of concurrency, students need to be careful with the timing in this activity. In constrast to the first activity where the distance is updated over a 1 second time period, in this activity the total distance needs to be updated in zero seconds.
In the third exercise, students will use object-level functions to implement simple collision detection.

Resources and Materials

- ⋆ Alice textbook
- ⋆ Computer suite with Alice 2.2 software
- ⋆ Desktop/laptop connected to data projector
- ⋆ Alice Tutorial: Object Functions and More Variables
- ⋆ Video tutorial: distance to function `https://www.youtube.com/watch?v=ORkkg9ZVdIA`

Differentiation

The third exercise includes an extension task in which students will use a local variable and a do together block to get a ball to bounce.

Lesson Closer (10 minutes)

In the lesson close, solutions to the three main tasks will be demonstrated. Students should be encouraged to share their solutions. If LanSchool or similar software is available, the student's solution can be projected to the rest of the class. Difficulties and challenges faced by the students should also be discussed at this stage. Students should also be given an opportunity to share any useful features that they have discovered in Alice.

Assessment

As a means of formative assessment the teacher should check on the progress of all students, marking off tasks as they are completed. For these tasks a simple checklist should suffice. Additional comments should also be recorded.

Lesson Comments

Students may find it difficult to complete all of the exercises in a double lesson. The teacher may either choose to extend the lesson or get the students to complete the tutorial for homework. The teacher should ensure that students complete the majority of exercises 1 and 3 in class, with exercise 2 a candidate for homework.

Lesson Plan: Practical Test 1

Subject	**Unit**
Year 11 Information Technology	Introduction to programming
Topic	**Lesson Duration/Date**
Practical Test 1	1 hour single lesson
Lesson Questions ⋆ How can different constructs be combined and used to implement programs?	**Objectives** ⋆ Students can use sequential and parallel blocks, simple loops, functions, methods and variables, to design and implement solutions to variety of pre-defined problems.
Introduction (5 minutes) In the lesson opener the teacher should explain that the lesson will be conducted under test conditions. The teacher should ensure that all students have access to the practical test question sheet, as well as the practical test program template. Students should be encouraged to read the questions carefully before the start programming. Instructions on how to submit their completed solutions should also be given at this time.	
Main Activity Task (40–50 minutes) The main activity for this lesson is Practical Test 1. Students are expected to work on this individually, with minimal support from the teacher.	
Resources and Materials ⋆ Alice textbook ⋆ Computer suite with Alice 2.2 software ⋆ Desktop/laptop connected to data projector ⋆ Practical Test 1 ⋆ Program solution template	**Differentiation** No extension questions are included in the practical test. The teacher may wish to include additional extension questions; however these additional questions should be weighted to ensure that students receive a reasonable grade by completing the main questions.
Lesson Closer (5 minutes) In the lesson close the teacher should ensure that all students have submitted their solutions.	
Assessment	**Lesson Comments**
The practical test represents a summative assessment task. The results should be used in determining the final grades of the students for this unit of work.	Students should submit their solutions, allowing the teacher to mark the test outside of class. The school's learning management system (LMS) should be used for this purpose. Feedback to the students should then be provided online.

Lesson Plan: Conditional Statements

Subject	**Unit**
Year 11 Information Technology	Introduction to programming
Topic	**Lesson Duration/Date**
Conditional Statements	2 hour double lesson
Lesson Questions	**Objectives**
⋆ How can a program be designed so that the behaviour is conditional? ⋆ How can conditional behaviour be implemented using if-then-else statements? ⋆ How can input functions be used to get information from the user of the program?	⋆ Students can design and implement programs using conditional statements ⋆ Students can use interactive commands to get input from the user

Introduction (20 minutes)

In the lesson opener the concept of conditional statements will be introduced. The teacher will also need to introduce the comparison operators. The teacher should also introduce one or more Boolean-valued functions – i.e. functions that return a true or false result.

To demonstrate conditional statements the teacher can use a simple lucky number guessing game as an example.

⋆ This example uses a conditional statement, variables and a comparison operator.
⋆ The example also uses an input function which enables the program to get information from the user. If input functions have not already been introduced the teacher should spend some additional time demonstrating their use.
⋆ The teacher should also demonstrate and explain the use of random numbers. Special note should be made of the fact that by default the random number function will return a real number. The teacher should demonstrate setting the integerOnly option to ensure that a whole number is generated.

Main Activity Task (70 minutes)

The main activity for this lesson will be a design and coding tasks described in Alice Tutorial: Conditional Statements. The aim of this task will be to develop a quiz game where the user is asked a series of questions and if they get the correct answer their score is incremented by one. The task has been broken down into a number steps which are described in detail in the worksheet. In the first step students are asked to create their world, however teacher's may choose to supply a prepared world in advance if they believe that too much time will be spent creating the world.
During this activity students will use conditional statements, variables, user input and functions. Students will be required to use the three different user input functions. Students will also be required to create and use new methods.

Resources and Materials

- ⋆ Alice textbook
- ⋆ Computer suite with Alice 2.2 software
- ⋆ Desktop/laptop connected to data projector
- ⋆ Alice Tutorial: conditional statements
- ⋆ Video tutorial: lucky number guessing game `http://www.youtube.com/watch?v=0alOULIwIDs`

Differentiation

Two extension tasks are included on the worksheet. These extension tasks require the student to develop new methods that implement animations in reaction to correct/incorrect answers. These extension tasks will give students an opportunity to explore the movement of objects in Alice.

Lesson Closer (10 minutes)

In the lesson close, a partial solution to the main task will be demonstrated. Students should be encouraged to share their solutions. If LanSchool or similar software is available, the student's solution can be projected to the rest of the class. Difficulties and challenges faced by the students should also be discussed at this stage. Students should also be given an opportunity to share any useful features that they have discovered in Alice.

Assessment

As a means of formative assessment the teacher should check on the progress of all students, marking off tasks as they are completed. For these tasks a simple checklist should suffice. Additional comments should also be recorded.

Lesson Comments

In this lesson students will be required to use a variety of functions. Many students will need assistance is using the functions correctly. Refreshing the score in step 5 will require the use of the string utility functions, which may require additional guidance – these functions should be demonstrated at the start of the class. Students may also have difficulties with scoping of variables; students should be encouraged to pass the values of variables as parameters.

Lesson Plan: Indefinite Loops

Subject	**Unit**
Year 11 Information Technology	Introduction to programming
Topic	**Lesson Duration/Date**
Indefinite Loops	2 hour double lesson
Lesson Questions	**Objectives**
⋆ How can a program be designed an action is performed repeatedly while a certain condition holds? ⋆ How can repeated behaviour be implemented using a while loop? ⋆ How can comparison operators, functions and logical operators be used in while loop conditions?	⋆ Students can design and implement programs using indefinite loop ⋆ Students can use functions and comparison operators to construct loop conditions ⋆ Students can define and use class methods to decompose a complex program into smaller parts

Introduction (20 minutes)

In the lesson opener the while (indefinite) loops will be introduced. The teacher should revisit simple loops and note that they are suitable when we know exactly how many times we want to repeat the behaviour, but we cannot always predict how many times we wish to repeat especially when we get inputs from outside of the program (e.g. from the user).
To illustrate the use of while loops the teacher can use guessing game similar to the one developed by the students, however in this case the user is allowed multiple guesses until they get the correct answer.

⋆ The teacher should initially step through a pseudo code design for the program (see lesson notes).
⋆ The program uses a while loop to repeatedly prompt the user for an answer.
⋆ Because we are using a while loop (pretest), an initial guess needs to be supplied that cannot be a correct answer. The teacher may wish to note that other languages provide loop constructs with post-tests (do until), which would be more suitable in this situation.
⋆ The teacher should take the opportunity to review comparison operators and logical operators as well as the random number function which will be used in the tutorial.

Main Activity Task (70 minutes)

The main activity for this lesson will be a design and coding tasks described in Alice Tutorial: Indefinite loops. The aim of this task will be to develop cat and mouse game in which the cat chases a mouse that is moving in a random direction. The task is broken down into a number steps, thus providing students good guidance on the top-level design.
During this activity students will be required to define and use a number of class level methods, i.e. methods defined for a particular object. The teacher should emphasive to students that this provides a useful way of breaking down a complex problem into smaller pieces. Students will be required to use a while loop in the main program. The loop condition will include a `distance to` function, as introduced in an earlier lesson.

Resources and Materials

- ⋆ Alice textbook
- ⋆ Computer suite with Alice 2.2 software
- ⋆ Desktop/laptop connected to data projector
- ⋆ Alice Tutorial: indefinite loops
- ⋆ Video tutorial: guessing game with retries `http://www.youtube.com/watch?v=V1_USARO2PE`
- ⋆ Sample solution: cat and mouse game `http://www.youtube.com/watch?v=C2FtawdWNvQ`

Differentiation

An extension activity is provided on the worksheet. In this extension activity students will use the probability of true function to provide some randomness to the outcome of the game. Students may also implement animations for the cat and mouse movement as well as the catch sequence once the main task is completed.

Lesson Closer (10 minutes)

In the lesson close, a partial solution to the main task will be demonstrated. Students should be encouraged to share their solutions. If LanSchool or similar software is available, the student's solution can be projected to the rest of the class. Difficulties and challenges faced by the students should also be discussed at this stage. Students should also be given an opportunity to share any useful features that they have discovered in Alice.

Assessment

As a means of formative assessment the teacher should check on the progress of all students, marking off tasks as they are completed. For these tasks a simple checklist should suffice. Additional comments should also be recorded.

Lesson Comments

This lesson provides a good opportunity for students to work in pairs. If the teacher chooses this option, they should ensure that the students are given sufficient guidance on the principles of pair programming and that the students share the driver and navigator roles.

Lesson Plan: Events and Event Handling

Subject Year 11 Information Technology	**Unit** Introduction to programming
Topic Events and Event Handling	**Lesson Duration/Date** 2 hour double lesson
Lesson Questions ⋆ What are events and actions in computer programs? ⋆ What are event handlers? ⋆ What events are supported in Alice? ⋆ How are event handlers developed in Alice?	**Objectives** ⋆ Students can design and implement event handling methods ⋆ Students can use events to develop interactive programs
Introduction (20 minutes) In the lesson opener events and event handlers will be introduced. The teacher should give students an overview of the events supported in Alice and demonstrate how event handler methods can be developed and called in Alice. TODO: add a video that describes the basic events in Alice and the writing of a simple event handler.	

Main Activity Task (70 minutes)

The main activity for this lesson will be two design and coding tasks described in Alice Tutorial: Events and Event Handling. In the first task students will use a mouse click event to turn a light switch on or off. In this example students will also need to create an object variable – the teacher may want to include a review of object variables (and other variables) at the start of the lesson.
In the second task students will use a variety of different events in developing a space world. Teachers may wish to demonstrate the completed code to the students before they attempt the exercise so they have a clear idea of what they need to develop.

Resources and Materials	**Differentiation**
⋆ Alice textbook ⋆ Computer suite with Alice 2.2 software ⋆ Desktop/laptop connected to data projector ⋆ Alice Tutorial: events and event handling	Two extension activities are provided that continue the second exercise. The first extension requires students to use a while world is running event. For the second extension activity students will develop simple collision detection.

Lesson Closer (10 minutes)

In the lesson close, a partial solution to the main task will be demonstrated. Students should be encouraged to share their solutions. If LanSchool or similar software is available, the student's solution can be projected to the rest of the class. Difficulties and challenges faced by the students should also be discussed at this stage. Students should also be given an opportunity to share any useful features that they have discovered in Alice.

Assessment	**Lesson Comments**
As a means of formative assessment the teacher should check on the progress of all students, marking off tasks as they are completed. For these tasks a simple checklist should suffice. Additional comments should also be recorded.	Students should be exposed to a variety of events during this lesson. In this lesson students will be presented with quite advanced concepts – students should be comfortable with the previous concepts, in particular creating methods, before they undertake this lesson.

Lesson Plan: Practical Test 2

<table>
<tr><td>Subject
Year 11 Information Technology</td><td>Unit
Introduction to programming</td></tr>
<tr><td>Topic
Practical Test 2</td><td>Lesson Duration/Date
1 hour single lesson</td></tr>
<tr><td>Lesson Questions
⋆ How can selection, iteration, user interaction and other functions be used to design and implement programs?</td><td>Objectives
⋆ Students can use if statements, user interaction, random numbers and while loops to design and implement solutions pre-defined problems.</td></tr>
<tr><td colspan="2">Introduction (5 minutes)
In the lesson opener the teacher should explain that the lesson will be conducted under test conditions. The teacher should ensure that all students have access to the practical test question sheet. Students should be encouraged to read the questions carefully before the start programming. Instructions on how to submit their completed solutions should also be given at this time.</td></tr>
<tr><td colspan="2">Main Activity Task (40–50 minutes)
The main activity for this lesson is Practical Test 2. Students are expected to work on this individually, with minimal support from the teacher.</td></tr>
<tr><td>Resources and Materials
⋆ Alice textbook
⋆ Computer suite with Alice 2.2 software
⋆ Desktop/laptop connected to data projector
⋆ Practical Test 2</td><td>Differentiation
No extension questions are included in the practical test. The teacher may wish to include additional extension questions; however these additional questions should be weighted to ensure that students receive a reasonable grade by completing the main questions.</td></tr>
<tr><td colspan="2">Lesson Closer (5 minutes)
In the lesson close the teacher should ensure that all students have submitted their solutions.</td></tr>
<tr><td>Assessment</td><td>Lesson Comments</td></tr>
<tr><td>The practical test represents a summative assessment task. The results should be used in determining the final grades of the students for this unit of work.</td><td>Students should submit their solutions, allowing the teacher to mark the test outside of class. The school's learning management system (LMS) should be used for this purpose. Feedback to the students should then be provided online.</td></tr>
</table>

References

[1] LANSchool. `http://www.lanschool.com/`. Accessed 26th April 2011.

[2] Moodle. `http://moodle.org/`. Accessed 26th April 2011.

[3] L. J Barker and W. Aspray. The state of research on girls and IT. *Women and information technology: Research on underrepresentation*, pages 3–54, 2006.

[4] C. Bishop-Clark, J. Courte, and E. V Howard. Programming in pairs with alice to improve confidence, enjoyment, and achievement. *Journal of educational computing research*, 34(2):213–228, 2006.

[5] J. Chao. Student project collaboration using wikis. 2007.

[6] S. Cooper, W. Dann, and R. Pausch. Alice: a 3-D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges*, volume 15, pages 107–116. Consortium for Computing Sciences in Colleges, 2000.

[7] Curverider Limited. elgg. `http://www.elgg.org/index.php`. Accessed 26th April 2011.

[8] W.P. Dann, S. Cooper, and R. Pausch. *Learning to Program with Alice*. Prentice Hall, 2006.

[9] Edmodo. Secure social learning network for students and teachers. `http://www.edmodo.com/`. Accessed 26th April 2011.

[10] S. P Ferris and H. Wilder. Uses and potentials of wikis in the classroom. *Innovate*, 2(5), 2006.

[11] N. D Fleming. I'm different; not dumb. modes of presentation (VARK) in the tertiary classroom. In *Research and Development in Higher Education, Proceedings of the 1995 Annual Conference of the Higher Education and Research Development Society of Australasia (HERDSA), HERDSA*, volume 18, page 308–313, 1995.

[12] R. Godwin-Jones. Emerging technologies: Blogs and wikis: Environments for On-Line collaboration. *Language, Learning & Technology*, 7(2), 2003.

[13] L. Grant. Using wikis in schools: A case study. *Retrieved March*, 2008.

[14] E. V Howard, D. Evans, J. Courte, and C. Bishop-Clark. A qualitative look at alice and pair-programming. In *Proceedings of ISECON 2006*, 2006.

[15] C. Kelleher, R. Pausch, and S. Kiesler. Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1455–1464. ACM, 2007.

[16] D. A Kolb, R. E Boyatzis, and C. Mainemelis. Experiential learning theory: Previous research and new directions. *Perspectives on thinking, learning, and cognitive styles. The educational psychology series*, page 227–247, 2001.

[17] B. Moskal, D. Lurie, and S. Cooper. Evaluating the effectiveness of a new instructional approach. In *ACM SIGCSE Bulletin*, volume 36, pages 75–79. ACM, 2004.

[18] K. R Parker and J. T Chao. Wiki as a teaching tool. *Interdisciplinary Journal of Knowledge and Learning Objects*, 3:57–72, 2007.

[19] SACE Board of SA. Sace stage 1 information technology. http://www.sace.sa.edu.au/subjects/stage-1/business-enterprise-and-technology/information-technology. Accessed 7th April 2011.

[20] J. S Safran. Supporting students with asperger's syndrome. *Teaching Exceptional Children*, 2002.

[21] J. G Sharp, J. Byrne, and R. Bowker. The trouble with VAK. *Educational Futures*, 1:76–93, 2007.

[22] T. C Wang, W. H Mei, S. L Lin, S. K Chiu, and J. M.C Lin. Teaching programming concepts to high school students with alice. In *Frontiers in Education Conference, 2009. FIE'09. 39th IEEE*, pages 1–6. IEEE, 2009.

A SACE Stage 1 IT

The SACE Stage 1 IT curriculum lists the following Key Question and and Concepts under the design section:

How is an application program designed?

- A program is designed by considering the inputs and the processing required to generate desired outputs.
- An algorithm is a series of steps written in sequence to define the solution to a problem. It can be expressed in pseudo-code.
- Recommended practice and efficient design involve the use of modules (procedures and functions).
- Procedures/modules are used to break a problem into manageable parts. A procedure is a group of statements that logically belong together and alter the state of the system.
- Structure charts indicate the order in which the procedures of a program are executed.

Note that this represents are relatively dated procedural-based model of application programming. Whilst procedural-based programming languages are still prevalent, since the 1990s there has been a growing shift towards object-oriented paradigms. The Alice programming environment used in this unit of work is based on an object-oriented paradigm. However the basic ideas of breaking the program into more manageable pieces is still a major concern of object-oriented design, so most of the stated goals are addressed in this unit of work.

The SACE Stage 1 IT curriculum lists the following Key Questions and Concepts under the development section:

How is the concept of variables fundamental to programming a computer-based application?

- A variable is a name assigned to a storage space in a computer.
- A value of a variable can be set or input or a result of a calculation.
- A variable is assigned a data type, such as string, integer, or floating point.
- A variable can be global or local.
- A value of a variable can be incremented, and the resulting value can be tested against a condition that responds with an action.
- A value of a variable can be input and output in various forms (e.g. text, object movement, media elements).

How is an application program developed?

- All computer programs use the control pseudo-code. structures of
  - sequence
  - selection (IF–THEN–ELSE, nested IF, CASE)

  - iteration (fixed, post-test, and pre-test loops).
- Data can be manipulated by using built-in functions (e.g. random, date, round, integer).

How is the design of an application program tested and how are errors resolved, if necessary?

- The types of errors (syntax, logic, execution) that occur in programming are identified in terms of their effect on the running of the program.
- A desk-check is used to test the validity of an algorithm.
- There are methods to reduce the occurrence of errors (e.g. tracing, debugging, flagging).

The first question relates to variables. In this unit of work there is very little explicit focus on variables. Object variables (attributes) will be accessed/updated via method calls, so will not be directly observed by students. Within a object oriented programming paradigm there is little call for global variables. As a result students will not encounter variables until late in the course where classes and objects are introduced.

The second question will be addressed throughout the course, with these language constructs firstly introduced on their own, then combined with other constructs in order to develop complex programs.

The final question, relating to testing and debugging, will be addressed throughout the course. However the Alice programming environment largely eliminates syntax errors (programs are constructed by drag and drop and menus rather than by typing). The main focus of this unit will be on eliminating logical errors, which are typically the most challenging errors to detect anyway (compilers and static analysis tools do a reasonable job of find syntax and runtime errors).

The Social Responsibility section of the Key Questions and Concepts lists the following:

What are the responsibilities of an application programmer?

- Recommended practices and conventions include using
  - comments within the code
  - an efficient design
  - a logical hierarchical folder structure.

What is the impact of application software on society?

- Application software
  - can be written, customised, or re- customised for specific purposes
  - has changed work practices, procedures, and decision-making processes
  - can be distributed commercially or as shareware or freeware, or may be modified and distributed further under open source conditions or a Creative Commons licence
  - is protected by copyright and, in some cases, other intellectual property rights, and may be subject to social, legal, and cultural practices and values
  - can be written for malicious and unethical purposes (e.g. viruses, spyware, adware, keystroke loggers).

The first question here is better answered under the design and development sections. A better question perhaps is what is an applications programmer? Student's should come to appreciate that applications programming can cover a whole range of areas, from games, social media, business, mining, defence, aviation etc and that developing programs requires a wide variety of skills beyond simply coding.

The second question could be answered within the context of a unit on Computer Ethics, which should be part of a broader IT curriculum rather than just focusing on applications programming. Neither of these questions are currently addressed explicitly in this unit. However it would be straightforward to include some short presentations and videos throughout the unit the discuss aspects of these concepts.