
INTRODUCTION TO COMPUTER PROGRAMMING

Program: Stage 5 Information Technology
Unit: Introduction to Computer Programming
Status: Complete
Revision: 4 (April 2007)
Author: Richard Laugesen
Contact: richard@tinyrock.com
Website: <http://tinyrock.com>

Copyright 2007 Richard Laugesen

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.



INTRODUCTION TO UNIT

This unit is part of the Stage 5 Information Technology curriculum at an Australian High School. It is intended to be delivered as the third of four units to year 10 students.

It has been designed to satisfy the objectives and requirements outlined on the previous page some of which are unique to this school. For this reason, and the intrinsic difficulties in teaching introductory programming, the software development environment Alice has been chosen rather than a more traditional programming language.

The unit is structured into explicit lesson plans containing all required content and activities; students *should* follow the lessons in sequence but own pace. Each concept is delivered via a built in tutorial and then backed up by a concrete activity. The activities are purposely simple so that all students may experience success, but easily extended so more gifted students may continue to feel challenged. Students should upload each completed activity for checking by their teacher. Teachers may assume the role of facilitators since all required content is contained within the Alice environment and the unit is designed for a constructivist learning approach. You may find that students drift from the lesson plan into programming tangents, this is great and should be encouraged!

The final assessment task hinges on the idea of using programming as an expression of creativity, and the marking scheme is therefore subjective.

INTRODUCTION TO ALICE

Alice is a piece of software which allows students to create **virtual worlds** containing complex scenes of many objects. The objects may then be programmed to behave in interesting and fun ways. For example a cat object could be programmed to generate a purring sound when clicked upon. The programming approach is to drag and click colourful blocks representing conventional programming concepts and then enter values to fine tune the block's behaviour. The end result is a computer program which animates a virtual scene under the students control and responds to inputs.

Alice is **free** software created by Carnegie Mellon University and backed by stacks of research suggesting that it is an excellent first exposure to programming.

Alice may be downloaded from alice.org

OBJECTIVES

- ❑ Introduce students to the concept of programming a computer.
- ❑ Maintain curiosity and engagement through an innovative curriculum.
- ❑ Create a few simple pieces of software.
- ❑ Encourage students to take further IT studies.
- ❑ Provide a framework within which students may demonstrate their ability to be self directed.

REQUIREMENTS

- ❑ Non-technical students with limited attention span and motivation.
- ❑ Limited critical thinking or mathematical skills.
- ❑ No programming or command line experience.
- ❑ Teachers with limited programming experience.
- ❑ Timeframe of 10 hours including an assessable project.
- ❑ Assume a constructivist learning approach with teachers acting as facilitators.

USEFUL RESOURCES

- ❑ Alice Website – <http://www.alice.org>
- ❑ Tutorials by Richard Baldwin – <http://www.dickbaldwin.com/tocalice.htm>
- ❑ An Alice textbook – <http://www.aliceprogramming.net>
- ❑ Alice community forum – <http://www.alice.org/community>

SCOPE AND SEQUENCE

Lesson	Task	Teacher Strategy	Register
1	Tutorial 1 – Alice interface and basic techniques	Main role is to facilitate student understanding in all lessons	
2	Activity 1 – Single object moving around a virtual world		
3	Tutorial 2 – More complex techniques		
4	Activity 2 – Two interacting objects		
5	Tutorial 3 – Adding sound and complex ordering	Provide students with additional If/Else resource (Appendix 1)	
6	Activity 3 – A slightly more intelligent object	Provide students with additional If/Else resource (Appendix 1)	
7	Individual Project	Provide students with assessment task and explain expectations (Appendix 2)	
8	Individual Project		
9	Individual Project		
10	Individual Project	Students complete and submit their projects for marking. Keen students could present their project to the class.	

* Note: Extension lessons for fast moving students are in Appendix 3/4

LESSON 1

Title: First Tutorial

Description: Become familiar with the Alice interface and learn basic techniques.

Tasks:

1. Complete *Tutorial 4*.
2. Discover the possibilities with Alice by experimenting with examples.
3. Create your first practice scene.

LESSON 2

Title: First Activity

Description: Create a simple project - Single object moving around a virtual world.

Useful Blocks: Movement , Do Together, Loop

Tasks:

1. Complete *Tutorial 1*.
2. Start a new project.
3. Save the project to the H: drive as *project1*.
4. Add an object to your scene.
5. Move the object 5 metres to the left and then 5 metres to the right.
6. Move the object up 1 metre.
7. Move the object down 1 metre but rotating at the same time.
8. Make the object jump up and down 10 times.
9. Save the project to the H: drive as *activity1*.
10. Submit your *activity1* file to your teacher.

LESSON 3

Title: Second Tutorial

Description: Become familiar with more complex techniques.

Tasks:

1. Complete *Tutorial 2*.
2. Play with the concepts you just learnt by modifying the examples.

LESSON 4

Title: Second Activity

Description: Create a more complex project - Two interacting objects.

Useful Blocks: Speech

Tasks:

1. Start a new project.
2. Save the project to the H: drive as *project2*.
3. Add two objects to your scene separated by a large distance.
4. Move one object so it is 1 metre from the other object.
5. Make the two objects have a conversation using speech bubbles.
6. Make the two objects dance around each other.
7. Save the project to the H: drive as *activity2*.
8. Submit your *activity2* file to your teacher.

LESSON 5

Title: Third Tutorial

Description: Become familiar with adding sound and complex ordering.

Tasks:

1. Complete *Tutorial 3*.
2. Play with the concepts you just learnt by modifying the examples.

LESSON 6

Title: Third Activity

Description: Create a more interesting project – A slightly more intelligent object.

Useful Blocks: Sound, Wait, If/Else

Tasks:

1. Open your *project2* file from the H: drive.
2. Make each object generate a unique sound when clicked on.
3. Use the If/Else block to make one object run away screaming from the other object (See the *How to use If/Else Block* section).
4. Save the project to the H: drive as *activity3*.
5. Submit your *activity3* file to your teacher.

LESSON 7

Title: Individual Project

Description: Complete an individual programming project.

Tasks:

1. Read the task assessment description and marking scheme.
2. Complete your project in this and the remaining lessons.
3. Save your project as *ProgrammingProject*.
4. Submit your project file for marking.

APPENDIX 1 - HOW TO USE AN IF/ELSE BLOCK

An *If/Else block* will make your object do something if something else is true. For example, a rabbit could double in size **if** it is shorter than a penguin.

1. Drag an *If/Else* block from the bottom of Alice onto the editor area.
2. Choose *true* for the initial condition.
3. Click on *functions* in the details area to see a list of possible conditions.
4. Replace the initial *true* condition with a condition from the functions list by dragging it onto your If/Else block.
5. Now add actions you want to happen if the condition is true or false by dragging them into the blank sections in your If/Else block.

APPENDIX 2 - INDIVIDUAL PROJECT

Unit: Introduction to Computer Programming

Due Date:

Total Possible Marks: 16

Description of Task:

- Must tell a story; it could be a comedy, thriller, or romance.
- Minimum of 3 characters, 3 scenes, and 4 props.
- Must include dialog between characters.
- Must use the concepts: Movement, Speech, Sound, Do Together, Loop

Marking Scheme:

<input type="checkbox"/> Includes minimum requirements	4	3	2	1	0
<input type="checkbox"/> Quality of Story	4	3	2	1	0
<input type="checkbox"/> Quality of Programming	4	3	2	1	0
<input type="checkbox"/> Quality of Animation	4	3	2	1	0

APPENDIX 3 - EXTENSION LESSON 1

Title: Take Control

Description: Create a more interesting project – A slightly more responsive object.

Useful Blocks: If/Else, While, Events

Tasks:

1. Start a new project.
2. Save the project to the H: drive as *extension*.
3. Add an object to your scene.
4. Make the object move left 1 metre if the left arrow key is pressed.
5. Make the object move right 1 metre if the right arrow key is pressed.
6. Make the object move forward 1 metre if the up arrow key is pressed.
7. Make the object move backwards 1 metre if the down arrow key is pressed.
8. Make the object jump if the space bar is pressed.
9. Add another object to the scene – such as a box.
10. Make the first object move backwards 2 metres if it is within 1 metre of the second object.
11. Now press play and move your object around the scene by pressing the arrow keys, drive it into the box and see if it behaves as you expect.
12. Save the project to the H: drive as *extension*.
13. Submit your extension file to your teacher.

APPENDIX 4 - EXTENSION LESSON 2

Title: Discovering Ruby

Description: Experience a more traditional programming language.

Tasks:

1. Visit this site: <http://tryruby.hobix.com/>
2. Read carefully and follow all instructions.

More Information:

- <http://www.ruby-lang.org/en/documentation/quickstart/>
- <http://pine.fm/LearnToProgram/>
- <http://www.ruby-doc.org/docs/ProgrammingRuby/>
- <http://hacketyhack.net/>
- <http://poignantguide.net/ruby/>