Laboratory 2

Objective
In this week’s laboratory we will investigate the use of variables in computer software. We will create and manipulate the three main types (numbers, text and boolean) of variables. We will look at the main functions provided by Scribble to provide input and output.

The aim of the laboratory is to help you:
1. Learn how to create and use test plans to test software applications
2. Learn how to apply debugging strategies to find and fix bugs in Scribble applications
3. Appreciate the value of good external software documentation.

Required resources
To perform this laboratory at Monash:
• You will need a Windows-based PC in a Monash student laboratory
• Your authcate credentials to login to the computer and to the unit-web site on Moodle

To perform this laboratory at home on your own computer:
• A working copy of Scribble
• Your authcate credentials to login to the computer and to the unit-web site on Moodle
• The Adobe Flash application (or a similar package) to run Adobe Shockwave (.swf) files.

Tasks (for on-campus students)
1. Alien Approach game play. Download a "clean" copy of the Alien Approach game that was shown in the the week 1 lecture. Remember this game - implemented in Flash - was created with a "block" style coding environment similar to the one we will use. It’s only a demo system but it does show that complex software can be created using this approach. Play the game for a few minutes - it's kind of fun for a while, then the novelty wears off a little :-).

   1.1. Write a description of the game and how it works. Just one or two paragraphs.
1.2. Create a description (more technical this time - use a diagram or two) of the way you imagine the software is working.

1.3. Once you have done this your tutor will talk for a few moments about the different approaches that people in the class have taken to conceptualising the architecture of the software, and about software architecture.

2. Developing a test plan for Alien Approach. With a partner develop a test plan than might be used by the makers of the software package if they were going to release it commercially and wanted to make sure there were no bugs.

3. Finding bugs in Alien Approach. On the unit web site you’ll find links to three different versions of the Alien Approach game that have been doctored to include some bugs. Your tutor will randomly allocate you a version to work with (A, B or C). It will be a different version to the one allocated to your partner:
   - Version A
   - Version B
   - Version C

3.1. Download the version of the game you have been allocated. Use your test plan - on your own - to test the application. Note any bugs you find and make suggestions as to which place in the software architecture you think the bugs will be found and how they might be fixed.

3.2. Now, do that again, but this time use the version of the game that your partner used.

3.3. Compare your results. Did you find the same bugs. Did you make similar or different suggestions about how to fix the problems?

3.4. Once everybody has finished your tutor will lead the class in a discussion. Reflect on:
   - The value of detail is in having a repeatable, comprehensive test plan.
   - The value of conceptualisation of the software architecture (as having a well designed architecture) is to finding and locating bugs.

At the end of the week a list of the bugs introduced to the various versions of the software will be released on the unit web site - take the time to compare the list you came up with with the "real" list.

4. Debugging Scribble Applications. For the next series of exercises read the outline of the problem and download the associated Scribble application. Open the application and try to identify and fix the bug. These tasks have been taken from the Debug it! exercises developed for Scratch by Karen Brennan. For each task you should follow the following procedure:

   1. Run the program
   2. Verify an error exists
   3. Find the problem in the code
   4. Fix the problem in the code
   5. Go back to the first step 1.

4.1. Application 1. Stephanie wants the cat to start in the middle of the stage, then move across the stage and grow. It works the first time she clicks the green flag – but not when she clicks it again! What’s going on?
4.2. **Application 2.** Mitch wants his cat to dance to some music. But the cat is dancing after the music is over! What’s going on?

4.3. **Application 3.** Karen wants to control the cat’s x-position with the keyboard: right arrow moves the cat right, left arrow moves the cat left. She also wants the cat to say if it’s on the right side or the left side depending on its x-position. The cat’s moving, but not saying its position correctly! What’s going on?

4.4. **Application 4.** Todd wants his cat to jump up and down, and then move across the screen. But the cat’s hopping across the screen! What’s going on?

5. That’s it - with about 30 minutes to go in the class your tutor will distribute the in-class test for this laboratory. This is worth 1% of your mark for the unit. It should only take 15 to 20 minutes to complete. You must work on it on your own. Give your answer sheet back to your tutor at the end of the class. You will get the result next week (don’t worry, it’s not hard).

### Tasks (for off-campus students)

1. **Alien Approach game play.** Download a "clean" copy of the Alien Approach game that was shown in the the week 1 lecture. Remember this game - implemented in Flash - was created with a "block" style coding environment similar to the one we will use. It’s only a demo system but it does show that complex software can be created using this approach. Play the game for a few minutes - it’s kind of fun for a while, then the novelty wears off a little :-).

1.1. Write a description of the game and how it works. Just one or two paragraphs.

1.2. Create a description (more technical this time - use a diagram or two) of the way you imagine the software is working.

2. **Developing a test plan for Alien Approach.** Develop a test plan than might be used by the makers of the software package if they were going to release it commercially and wanted to make sure there were no bugs.

3. **Finding bugs in Alien Approach.** On the unit web site you’ll find links to three different versions of the Alien Approach game that have been doctored to include some bugs.

   - **Version A**
   - **Version B**
   - **Version C**

3.1. Download the each of the available versions of the game and use your test plan to conduct a test of each of them. Note any bugs you find and make suggestions as to which place in the software architecture you think the bugs will be found and how they might be fixed.

3.2. Post your results in the off-campus students discussion forum. Compare the bugs you found with those reported by others. Did you find the same bugs. Did you make similar or different suggestions about how to fix the problems?

3.3. Before finishing reflect on the following points. Feel free to add comments about them to the posting you make on the discussion forum.

   - The value of detail is in having a repeatable, comprehensive test plan.
   - The value of conceptualisation of the software architecture (as having a well designed architecture) is to finding and locating bugs.
At the end of the week a list of the bugs introduced to the various versions of the software will be released on the unit web site - take the time to compare the list you came up with with the "real" list.

4. **Debugging Scribble Applications.** For the next series of exercises read the outline of the problem and download the associated Scribble application. Open the application and try to identify and fix the bug. These tasks have been taken from the *Debug it!* exercises developed for Scratch by Karen Brennan. For each task you should follow the following procedure:

1. Run the program
2. Verify an error exists
3. Find the problem in the code
4. Fix the problem in the code
5. Go back to the first step

4.1. **Application 1.** Stephanie wants the cat to start in the middle of the stage, then move across the stage and grow. It works the first time she clicks the green flag – but not when she clicks it again! What's going on?

4.2. **Application 2.** Mitch wants his cat to dance to some music. But the cat is dancing after the music is over! What's going on?

4.3. **Application 3.** Karen wants to control the cat's x-position with the keyboard: right arrow moves the cat right, left arrow moves the cat left. She also wants the cat to say if it's on the right side or the left side depending on its x-position. The cat's moving, but not saying its position correctly! What's going on?

4.4. **Application 4.** Todd wants his cat to jump up and down, and then move across the screen. But the cat's hopping across the screen! What's going on?

5. On the “study guide” tab of the unit-web site for this week you will find a link to this week's on-line test for the semester. This is worth 1% of your mark for the unit. It should only take 15 to 20 minutes to complete. You must work on it on your own. The test will be available for you to do for 1 week (you won’t be able to access it after midnight on Sunday). If you have any problem accessing the test please send Peter O’Donnell (peter.odonnell@monash.edu) an email.

**Further reading and information**

