Infant K1 Coding Pack: 1-4 Spot the Bug

Published Thursday, April 17th, 2014 | By Dave Catlin, Kate Hudson and Alan Coode

A game where students decide whether programs will work or whether they contain bugs.

Subjects	Age	Roamer Expertise	Student Grouping	Lesson Time	Availability
Computing Mathematics	Year 2 Year 1				

Description

Students work in teams. Each team writes three programs, which they test with Roamer. They then change one or two of the programs so they deliberately contain bugs. They pass their programs to another team. Each team fills in a scorecard identifying the bugs in the other teams programs. In a whole class session each team reviews the programs of all the other teams and marks which programs they think are "buggy". Teams score a point for each program they correctly identify as "bug-free" or "buggy".. The team that scores the most points wins.

Objectives

Students have the opportunity to:

1. Reason and predict what a program will do
2. Learn the basic process of debugging a program
3. Improve their Roamer programming skills
4. Develop their code reading skills
5. Practice basic numeracy skills

Secondary Objectives

Students will have the opportunity to develop sustainable skills:

1. Decision making
2. Cooperation
3. Computational thinking

Vocabulary

New words and phrases:

1. Bug
2. Software Testing
3. Debugging
4. Code Reading

KS1 Computing Pack

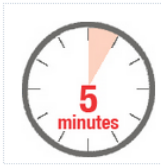
Index to KS1 Computing Pack Activities.

1– 4 Spot the Bug

Lesson Plan and Assessment

Preparation

1. Print out resources



Print these from the activity pdf.

1. Team Programming sheets
2. Team Score Cards
3. Game Rules

2. Prepare demo program

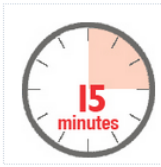
1. Download PowerPoint File
2. Set up projector or interactive whiteboard
3. Organise alternative the whiteboard

Activity

1. Explain programming bugs

1. Show the whole class the demo program written on the whiteboard
2. Engage the class in discussing the program:
   a. Is there anything wrong with the program?
   b. Can anyone say what is wrong?
   c. Could anyone re-write the program?
3. Discuss how they can find out if a program is correct?
   a. Test the program using Roamer
   b. Can anyone say what is wrong without using Roamer?
      i. Looking at the code
      ii. This is called code reading

2. Write the programs



1. Form class into buddy pairs
2. Instruct the students to write 3 programs:
   a. Each program should be at least 5 instructions long
   b. They should test the programs with Roamer
   c. There should be no bugs
3. Monitor class:
   a. Remind students how to use traffic lights if they need help

1– 4 Spot the Bug

3. Add bugs to the program

1. Explain the game to the children
   a. They should change one or two of their programs so they do not work
   b. The other teams are going to look for the bugs
   c. The other teams score points for every correct and buggy program they find
   d. So make the bugs hard to find
2. Hand out the program sheets
   a. Write your team name on the sheet
   b. Write your programs on the program sheets

4. Find the bugs

1. Handout the scorecards
2. Each team passes their programming sheet to another team
3. Add the competitor's team name to your scorecard
4. Find the bugs in the competitor's programming sheet
5. Record the bugs for your scorecard
6. Repeat the process until each team has reviewed all the programming sheets

Assessment

1. Results time

1. Ask all the teams did program A from team 1 contain a bug.
   a. Team 1 reveal the answer.
   b. The teams give themselves a point if they got it right.
   c. Repeat the process for the other two programs.
2. Repeat the process for the other teams.
3. Each team should add up their total scores.
   a. The teams should announce their scores..
   b. Declare the winner.

2. Review the paperwork

After the lesson review programming sheets and scorecards:

1. How complex were the programs written by the teams?
2. How clever were the bugs?
3. How good were the teams at finding the bugs?

1– 4 Spot the Bug

Teacher's Notes

Subject Comments

There is an Activity on debugging software in all three primary school Computing Packs. Testing software is a major part of computer programming. Despite the simplicity of the Roamer programming language, it is still possible to learn how to test and debug programs. Software testing is an orderly process, which the students will grasp by the end of the Pack 3. In this pack we introduce "code reading".

Students will get more value from Roamer when their programming skills improve. Roamer's talking help feature stops students making a mistake entering a command; Roamer does not accept the command and it tells them the correct syntax. However, as the more advanced Activities will show, you can program the correct syntax, but still create a bug. So there are two possible problems:

1. A syntax bug such as Forward, Forward.
2. The program not doing the right thing: for example Forward instead of Backward

In this Activity we concentrate on syntax bugs.

At this stage, students are only starting to learn how to program the robot. They often write one instruction then test it. So they may program Roamer to go to A. Then figure out how much the robot should turn. They make sure they get that bit right, before they program the last step to go to B.

This is not how programmers test code. While students can write their programs using this simple instruction-by-instruction method, they test the code of their classmates by code reading several instructions.

Prior Knowledge

Students should:

1. Have at least a little experience programming Roamer's basic commands:
   a. Forward, Backward, Left and Right, Wait and if possible music.
   b. Use the abbreviations for these instructions: FD, BK, LT, RT, W and d for music (see Technical Help below).
2. Know how to write instructions (as shown above) or be able to draw symbols.

1– 4 Spot the Bug

Teacher's Notes

Practical Hints

You can buy Roamer Cards or make your own. If you decide to make your own RUG Members will find the Publisher Template in the **Free** Downloads useful.

Roamer Program Publisher Template

Handwriting Infant K1 Roamer Programs

Shop 1522-123 Roamer Instruction Cards

Technical Help

You will find suggestions for handwriting commands in the User Guides on the Training Site.

Handwriting Roamer Commands

Science of Learning

Our first solution to a problem is often wrong, but we use it to find a better answer. This is a standard problem-solving technique. Even mathematicians, normally so precise, have a technique called "iteration" based on this idea. They use a rough solution to find a better one, which they use to find an even better solution. Papert made debugging programs a natural part of Logo. For example, if a student's fails to get Roamer to turn from "A", and point towards "B", they can try again. They use the first result to decide whether to turn more or less. This goes beyond developing coding skills. Debugging and the spirit of trial and improvement are part of all Roamer Activities and a general problem-solving skill.

LOGO, Turtles and Roamer

Seymour Papert invented LOGO as a computer language for education. LOGO includes Turtle Graphics: commands used for controlling a Turtle robot. Roamer inherits the educational value of LOGO because it is a Turtle type robot and its programming language is a dialect of LOGO and Turtle Graphics.

References and Useful Links

BBC and Open University. (1983). Talking Turtle. *Horizon Documentary Series.*

Papert, S. (1980). *Mindstorms, Children Computers and Powerful Ideas.* New York: Basic Books.

Papert, S. (1993). *The Children's Machine.* New York: Basic Books.

1– 4 Spot the Bug

Standard Roamers

1520-401 Early Years Roamer (One Roamer per Group)

Demo

PowerPoint

Printed Resources from PDF

Team Programming sheets
Team Score Cards
Game Rules

1– 4 Spot the Bug

Names _____

Team Name

How to Write Instructions

 forward. FD

 backward BK

 left LT

 right RT

wait W

Write Your 3 Programs Below

Program 1

Program 2

Program 3

1– 4 Spot the Bug

Resources: Score Card

Names _____

Team Name _____

Mark your scorecard

Right ✔ Bug ✘

Team	Programs 1	2	3