Implementing Digital Technologies Masterclass

James Curran (james@groklearning.com)
Associate Professor, School of IT, University of Sydney
Director, National Computer Science School
CEO and Co-founder, Grok Learning

Vivian Li (viv@groklearning.com)
Engineer, Grok Learning

THE UNIVERSITY OF SYDNEY

NATIONAL COMPUTER SCIENCE SCHOOL

GROK LEARNING

Today's program

8:30 Lessons from Microsoft WeSpeakCode

Introducing 7,000 students to coding in one week

Andrew Coates, Developer Evangelist, Microsoft

9:20 Digital Technologies

An overview of where we are at, where we are heading & global connections

Julie King, Senior Project Officer Technologies, ACARA

10:00 Computational Thinking

Transforming learning spaces, engaging and empowering teachers & students through practical ideas, tools & strategies

Nikos Bogiannidis, Dean of Learning Technologies, Haileybury School

10:45 Morning Tea & Networking

GROK LEARNING

Today's program

11:15 Digital Technology Curriculum Masterclass

How to implement the new Digital Technologies Curriculum in your school

James Curran and Vivian Li

1:00 Networking Lunch

2:00 Digital Technology Curriculum Masterclass (cont.)

How to implement the new Digital Technologies Curriculum in your school

James Curran and Vivian Li

4:30 End of Masterclass

GROK LEARNING

Programming in Digital Technologies

- What you need to cover
- Platforms and languages
- Grok Learning / NCSS Challenge

GROK LEARNING

Some good reasons to teach x

- x is a different way of thinking or making
- x is tells us something about the world
- x is challenging, engaging and creative
- x is fundamental for understanding other disciplines
- x is a skill that takes years to master, so we should expose kids to it early

GROK LEARNING

Some bad reasons to teach x

- industry wants x
- there are jobs in x now
- the money is good in x
- x gives you a high ATAR
- x is in the exam
- ...

1

What you need to cover

GROK LEARNING

Aspects of programming languages

- visual *or* text programming
- object-oriented programming
- teaching (specialised) *or*
  real-world (general purpose)

Visual programming

- *mandatory* Year 3-6, maybe Year 7/8
- drag and drop elements to create code
- avoids syntax (and some semantic) errors

- unplugged: **flowcharts**, decision trees
- teaching: **Scratch**, Alice, Blockly, Lego NXT
- real-world: **Labview**

GROK LEARNING

Text programming

- *mandatory* by Year 8, maybe Year 6-7
- write code by hand
- more efficient and succinct than visual

- unplugged: **pseudocode**
- teaching: Actionscript, *Basic, **Python**, ...
- real-world: Actionscript, C++, C#, Java, Javascript, **Python**, Visual Basic

GROK LEARNING

Object-oriented (OO) programming

- *mandatory* by Year 10, maybe Year 8-9
- data and code organised as objects
  - objects model the real world
  - support classes, inheritance, and polymorphism

- unplugged: **the world is all objects**
- teaching: C#, Java, **Python**, VB(ish)
- real-world: C++, C#, Java, Javascript(ish), **Python**
- Scratch has objects, but isn't OO

GROK LEARNING

Specialised versus general purpose

- some languages are domain specific:
  - teaching (Scratch, Alice, Blockly, ABC, Blue)
  - game development (Game Maker, Unrealscript)
  - mathematics (Mathematica, MATLAB)
  - databases (SQL) and spreadsheets (Excel formula)
  - often lack common language features
- most languages are general purpose:
  - solve problems in a wide range of domains
  - solve specific domain problems less elegantly

Specialised versus general purpose

- language may be general, but platform not
- migrate specialised → general purpose
  - Javascript (now used on the backend as node.js)
  - Scratch (used for control on Raspberry Pi)
- general purpose → specialised libraries
  - Python: Django and others to hide SQL
  - Python: numpy to replace MATLAB
- Digital Technologies mandates general purpose by Year 8 to provide students with a practical programming tool

Implementation

4 Implement simple digital solutions as **visual programs** with algorithms involving **branching** (decisions) and **user input**

6 Implement digital solutions as simple visual programs involving branching, **iteration (repetition)**, and user input

Implementation

8 Implement and **modify programs** with **user interfaces** involving branching, iteration and **functions** in a **general-purpose programming language**

10 Implement **modular programs**, applying selected algorithms and **data structures** including using an **object-oriented programming language**

GROK LEARNING

Implementation

	Concepts	Programming	Test and debug
2		*Can do physical programming (e.g. Bee Bot)*	
4	Branching (decisions) and user input	Visual programming	
6	Iteration (repetition)	Visual programming	
8	User interfaces and functions	General purpose text programming	*In algorithms content descriptor*
10	Modularity, algorithms and data structures	Object-oriented programming	*In algorithms content descriptor*

Specification and Algorithms

2 Follow, describe and represent a **sequence of steps and decisions** (algorithms) needed to solve simple problems

4 **Define** simple problems, and describe and follow a **sequence of steps and decisions** (algorithms) needed to solve them

6 Define problems in terms of **data and functional requirements** drawing on previously solved problems

6 Design a **user interface** for a digital system

6 Design, modify and follow simple algorithms involving **sequences of steps**, **branching**, and **iteration**

GROK LEARNING

Specification and Algorithms

- Define and **decompose real-world problems** taking into account functional requirements and **economic, environmental**, **social**, **technical** and **usability** constraints
- Design the **user experience** of a digital system, **generating**, **evaluating** and communicating **alternative designs**
- Design algorithms represented **diagrammatically and in English**, and **trace algorithms** to predict output for a given input and to identify errors

Specification and Algorithms

10 Define and **decompose real-world problems** precisely, taking into account **functional and non-functional requirements** and including **interviewing stakeholders** to identify needs

10 Design the user experience of a digital system by **evaluating** alternative designs **against criteria** including **functionality**, **accessibility**, **usability**, and **aesthetics**

10 Design algorithms represented diagrammatically and in structured English and validate algorithms and programs through tracing and test cases

Specification and Algorithms

Describe problems	Follow/design algorithms	Design user interfaces
	Follow, describe and represent a sequence of steps and decisions	
Define simple problems	Describe/follow a sequence of steps and decisions (algorithms) needed to solve problems	
Define problems using data and functional requirements	Design/modify simple algorithms (also) involving iteration	Design a user interface
Decompose real-world problems; Consider economic, environmental, social, technical and usability constraints	Represent algorithms using diagrams and English; trace algorithms	Generate and evaluate alternative designs
Interviewing stakeholders to identify needs	Validate algorithms and programs through tracing and test cases	Evaluate designs against criteria: functionality, accessibility, usability, and aesthetics

2
4
6
8
10

2

Platforms and languages

Bee Bots

GROK LEARNING

- Bee bot!

GROK LEARNING

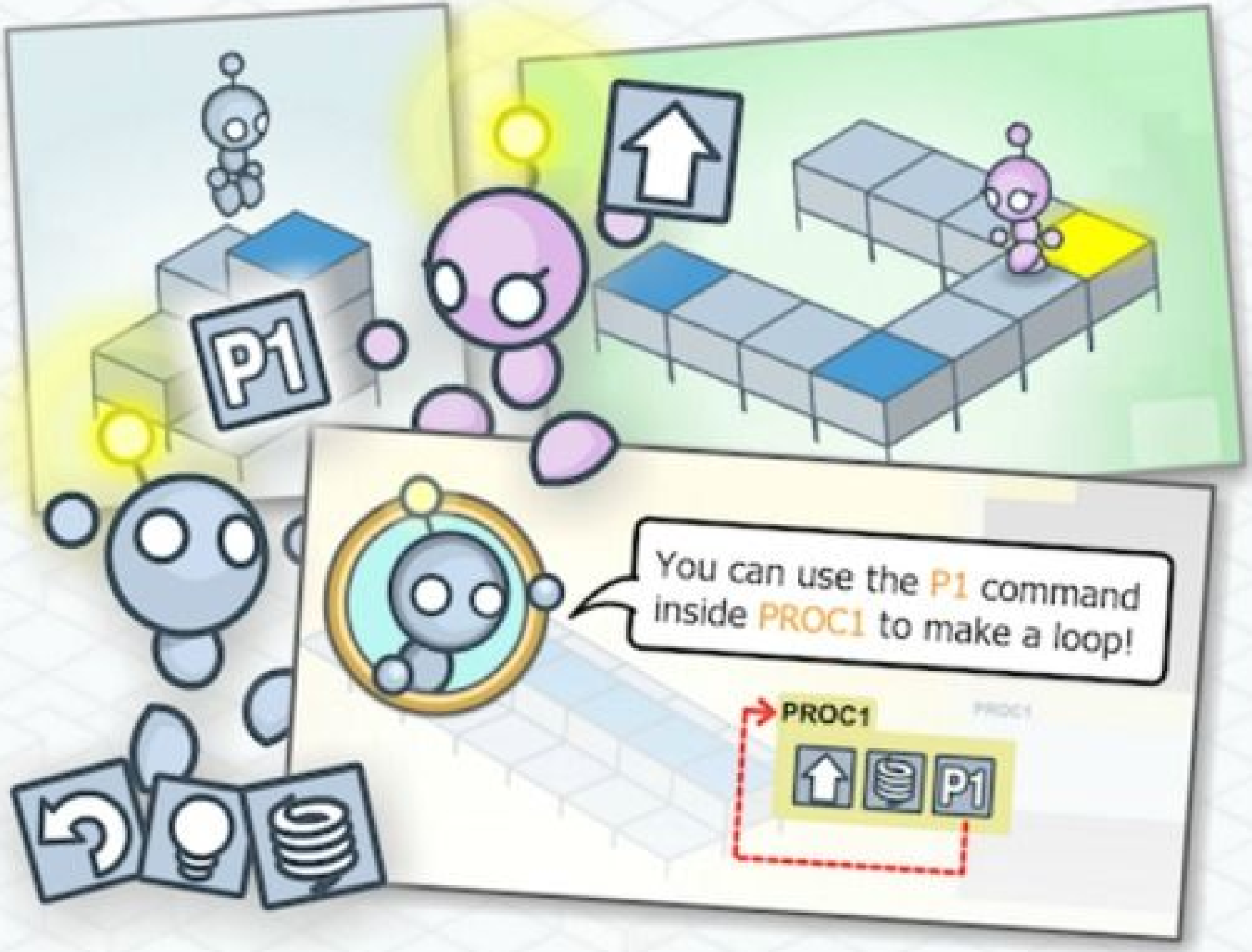
Cambodian Children's Trust

GROK
LEARNING

Lightbot (HoC activity)

lightbot.com/hocflash2014.html

GROK LEARNING


P1
You can use the P1 command inside PROC1 to make a loop!
PROC1
P1

Code.org and Hour of Code

code.org/learn

GROK LEARNING


move forward by 100 pixels
turn right by 90 degrees
Creativity powered by
Disney
INFINITY

Scratch

scratch.mit.edu

GROK LEARNING


design a holiday card
Click to explore!
animate your name
create a pong game
SCRATCH

3

Grok Learning

https://groklearning.com

GROK LEARNING

Representation

2 **Recognise and explore patterns** in data and **represent data** as pictures, symbols and diagrams

4 Recognise **different types of data** and explore how the **same data can be represented in different ways**

6 Examine how **whole numbers are used to represent all data** in digital systems

8 Investigate how digital systems **represent text, image and audio data in binary**

10 Analyse simple **compression of data** and how **content** data are **separated from presentation**

GROK LEARNING

Representation

	Representation	Types of data	Compression
2	Represent data as pictures, symbols and diagrams		
4	The same data can be represented in different ways	Different types of data	
6	How whole numbers are used to represent all data	All (simple) data: *types should be more complex*	
8	Represent data in binary	Text, image and audio	
10	Content vs. presentation: *documents are represented*	All data: *structured data*	Simple compression of data

GROK LEARNING

Collection and interpretation

2 **Collect**, **explore** and **sort** data, and use digital systems to **present** the data creatively

4 Collect, access and present **different types of data** using simple software to **create information** and **solve problems**

6 **Acquire, store** and **validate** different types of data, and use a range of software to **interpret and visualise data to create information**

GROK LEARNING

Collection and interpretation

8 Acquire data from a **range of sources** and **evaluate authenticity**, **accuracy** and **timeliness**

8 Analyse and visualise data using a **range of software** to create information, and **use structured data** to model objects or events

10 Develop techniques for acquiring, storing and validating **quantitative/qualitative data** from a range of sources, considering **privacy** and **security** requirements

10 Analyse and visualise data to create information and address complex problems, and **model processes**, **entities and their relationships** using **structured** data

GROK LEARNING

Collection and interpretation

	Collect	Organise / create	Visualise
2	Collect and explore data	Sort data	Present the data
4	Collect and access different types of data	Create information and solve problems	
6	Acquire, store and validate different types of data	Interpret data to create information	Visualise data to create information
8	Evaluate authenticity, accuracy and timeliness	Use structured data to model objects or events	Visualise data using a range of software
10	Validating quantitative and qualitative data; considering privacy and security	Model processes, entities and their relationships using structured data	Visualise data to create information and address complex problems

Specification and Algorithms

2 Follow, describe and represent a **sequence of steps and decisions** (algorithms) needed to solve simple problems

4 **Define** simple problems, and describe and follow a **sequence of steps and decisions** (algorithms) needed to solve them

6 Define problems in terms of **data and functional requirements** drawing on previously solved problems

6 Design a **user interface** for a digital system

6 Design, modify and follow simple algorithms involving **sequences of steps**, **branching**, and **iteration**

Specification and Algorithms

8 Define and **decompose real-world problems** taking into account functional requirements and **economic, environmental**, **social**, **technical** and **usability** constraints

8 Design the **user experience** of a digital system, **generating**, **evaluating** and communicating **alternative designs**

8 Design algorithms represented **diagrammatically and in English**, and **trace algorithms** to predict output for a given input and to identify errors

Specification and Algorithms

10 Define and **decompose real-world problems** precisely, taking into account **functional and non-functional requirements** and including **interviewing stakeholders** to identify needs

10 Design the user experience of a digital system by **evaluating** alternative designs **against criteria** including **functionality**, **accessibility**, **usability**, and **aesthetics**

10 Design algorithms represented diagrammatically and in structured English and validate algorithms and programs through tracing and test cases

GROK LEARNING

Specification and Algorithms

	Describe problems	Follow/design algorithms	Design user interfaces
2		Follow, describe and represent a sequence of steps and decisions	
4	Define simple problems	Describe/follow a sequence of steps and decisions (algorithms) needed to solve problems	
6	Define problems using data and functional requirements	Design/modify simple algorithms (also) involving iteration	Design a user interface
8	Decompose real-world problems; Consider economic, environmental, social, technical and usability constraints	Represent algorithms using diagrams and English; trace algorithms	Generate and evaluate alternative designs
10	Interviewing stakeholders to identify needs	Validate algorithms and programs through tracing and test cases	Evaluate designs against criteria: functionality, accessibility, usability, and aesthetics

Implementation

4 Implement simple digital solutions as **visual programs** with algorithms involving **branching** (decisions) and **user input**

6 Implement digital solutions as simple visual programs involving branching, **iteration (repetition)**, and user input

Implementation

8 Implement and **modify programs** with **user interfaces** involving branching, iteration and **functions** in a **general-purpose programming language**

10 Implement **modular programs**, applying selected algorithms and **data structures** including using an **object-oriented programming language**

GROK LEARNING

Implementation

	Concepts	Programming	Test and debug
2			
4	Branching (decisions) and user input	Visual programming	
6	Iteration (repetition)	Visual programming	
8	User interfaces and functions	General purpose text programming	*In algorithms content descriptor*
10	Modularity, algorithms and data structures	Object-oriented programming	*In algorithms content descriptor*

GROK LEARNING

Questions?

Find me at:
@drjamescurran / @groklearning
james@groklearning.com

GROK LEARNING