Learning Technology Research Group

- **Areas of Interest**
  1. Pedagogy
     - Vygotsky, Piaget etc.
  2. Methods and analysis
     - Activity Theory
     - Learning Metrics
     - Grounded Theory
     - Phenomenography
  3. Technology and Software
     - E-Assessment and Feedback
     - Learning Environments

- **People**
  - Graham Alsop
  - Alicia Campos
  - (Nick Fernando)
  - Maryam Kheir-Abadi
  - Dave Livingstone
  - Paul Neve
  - (Chris Tompsett)
Coherence

- The aspects are not separable: Pedagogy, Methods and Technology...
- Theory comes with baggage (whether it is an Educational or Research Approach)
- Pragmatic approach - use what best fits the problem
- Technology - needs to be useful
- Project focused
Improved Learning

Technology and Software

Methods and Analysis

Pedagogy

Improved Learning
Pedagogy

Plato (knowledge is innate)

Piaget (individual constructivism)

Vygotsky (social constructivism)

Lave (situated cognition)

Skinner (behaviourism)

We need to know a little about this 😊
Methods and Analysis

- Journal publications on:
  - Phenomenography
  - Grounded Theory
  - Communities of Practice

- Conference Papers on:
  - Action Research
  - Activity Theory

- Current Research:
  - Activity Systems
  - Learning Metrics
Why is teaching programming a problem?

- Lectures alone are inadequate
  - Schank (2001): a paper textbook represents a superior learning experience compared to a lecture

- Many pedagogic approaches, e.g.
  - Lectureless forms of delivery date back as far as Daly et al. (1979)
  - Constructivist approaches such as Wulf (2005) - tutor acts as guide on the side

- The short version: programming is a practical activity and any successful teaching approach must put the focus onto practice...

- ...so we have the “default” approach
  - Framing lecture followed by practical workshop
  - There is research that supports this approach e.g. Poindexter (2003)
  - Even Wulf concedes the need for framing lectures to set the scene and provide a framework for practical experimentation
The learning "curve" in computer programming

Time

Classic learning curve

‘Light Bulb’ model
The ideal workshop session and the "learning loop"

1. Student undertakes a practical workshop activity.
2. During the activity, the tutor offers feedback - this might be requested by the student, or volunteered spontaneously based on observations of the student's work.
3. Student responds to the feedback - either verbally or in their subsequent activity path.
4. Future feedback from the tutor is in turn influenced by the student's response.

The Learning Loop
British HE: Thereality

...outdated or badly configured equipment in computer labs...
British HE: The reality

...large cohorts...
British HE: The reality

...increased demand for distance and flexible learning...

(BBC 2010; Scottish Government 2011)
The research - understand how first year students learn to programme

The method - Activity Theory (Engestrom)

From the perspectives of the communities involved - all

Output - PhD submission - a new methodological approach to study using Activity Systems (Maryam Kheir-Abadi)

Direct input into first year modules
Methods and Analysis: Learning Metrics

- Using metrics generated by students in learning environments for adaptive pedagogy (Alicia Campos and Paul Neve)
  - From the perspective of the student's learning process
    - achievements, progress, effort, confidence and confusion
  - From the perspective of the learning content
    - Time to learn, rate of errors, rate of frustration, overall effectiveness
  - Patterns and signatures
    - Similarities between students

- The learning environments KUOLE and NoobLab gather these metrics and provide a platform for this research
Technology and Software: Learning Environments

- **KUOLE**
  - Interactive, immersive learning environment that combines static text content, multimedia and formative "quiz" style content

- **NoobLab**
  - Specialist environment for teaching programming
  - Presents both the "framing" content and an area where the student can practically engage with program code
  - Allows for the design of practical programming exercises, against which a student can test their code

- Both tools...
  - provide a platform for gathering and analysis of learning metrics
  - combine teaching delivery with the ability to inform course design and pedagogy
Creating String Variables and Displaying their Contents

Abstract:

1. Open the file `StringVariables.java` using your Java Editor. Compile and run the program. You should see the following message on the screen in the console box:

```
Welcome to Workbook 2!
Press any key to continue . . .
```

Look at the following lines of code that declare a string variable called `myMessage` and display its contents on the screen:

```java
String myMessage = "Welcome to Workbook 2";
System.out.println(myMessage);
```

1. Now add another line of code to the program to display the contents of the variable `anotherMessage` on the screen using `System.out.println`.
2. Now create another `String` variable containing any text you like, and display the contents of the variable on the screen using `System.out.println`.

EXERCISE 1

When you have completed this exercise:

Copy your code in the box provided below and click on 'Submit my answer'.

Note: You can submit your code as many times as you wish if you want to change it.
Practical Programming - Lesson 1

IF/ELSE - basic conditional statements

Up until now, our programs have followed a linear flow: they have started at the first line, progressed through the code line by line until they reach the end. They follow exactly the same order of execution every time the program runs.

Consider your home central heating system. Most likely, if it is a modern system, there will be some kind of embedded computer device that controls the temperature. One of the tasks this computer will have will be to monitor the current ambient temperature. If it gets too hot in the house, the computer will turn down or even turn off the heating system. If it gets too cold, it will turn on or turn up the heating system.

The operative word here is if and the if statement is one of the most fundamental constructs in any computer programming language. It is an example of a conditional construct, and allows us to code in different behaviours depending on the conditions that exist in your program at a given time.

Let’s take a look at an example:

```javascript
var myVariable = input("Enter a number");
if (myVariable == 12) println("myVariable is equal to 12.");
```

This program gets an input from the user, puts it into the variable number. Then, if the number is equal to 12 it displays a message.

**Quiz Question 14 (click your answer)**

What happens if the number is NOT equal to 12?

- Nothing - the program just ends.
- It displays a message saying that the number is not equal to 12.
- The program throws an error if the number is not 12.

Sorry! Your program did not produce the expected output. It passed 2 out of 3 tests. Check your work and try again!
The NoobLab environment gathers usage statistics from students. We anticipate that common patterns or signatures will emerge.

<table>
<thead>
<tr>
<th>Index</th>
<th>Timestamp</th>
<th>Action</th>
<th>Location</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>09:59:00 2012/02/06</td>
<td>RunStart</td>
<td>CI1152B:1:7</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>09:59:03 2012/02/06</td>
<td>RunUserInput</td>
<td>CI1152B:1:7</td>
<td>Paul</td>
</tr>
<tr>
<td>49</td>
<td>09:59:03 2012/02/06</td>
<td>RunSuccess</td>
<td>CI1152B:1:7</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>09:59:15 2012/02/06</td>
<td>TestStart</td>
<td>CI1152B:1:7:firstlf</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>09:59:15 2012/02/06</td>
<td>TestFailed</td>
<td>CI1152B:1:7</td>
<td>2/3</td>
</tr>
<tr>
<td>52</td>
<td>10:01:29 2012/02/06</td>
<td>RunStart</td>
<td>CI1152B:1:7</td>
<td>4</td>
</tr>
<tr>
<td>53</td>
<td>10:01:31 2012/02/06</td>
<td>RunUserInput</td>
<td>CI1152B:1:7</td>
<td>PAUL</td>
</tr>
<tr>
<td>54</td>
<td>10:01:31 2012/02/06</td>
<td>RunSuccess</td>
<td>CI1152B:1:7</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>10:01:32 2012/02/06</td>
<td>TestStart</td>
<td>CI1152B:1:7:firstlf</td>
<td>0</td>
</tr>
<tr>
<td>56</td>
<td>10:01:33 2012/02/06</td>
<td>TestPassed</td>
<td>CI1152B:1:7</td>
<td></td>
</tr>
</tbody>
</table>
Learning technology as a tool for informing pedagogy
Other Technology and Software:

- **Electronic Assessment**
  - The LTRG’s work has established KU as a leading research institution on the IMS Global Learning Consortium’s Question and Test Interoperability (QTI) standard:
    - Aqurate, Mathqurate, Spectatus and current project Uniqurate provide authoring tools for QTI e-assessment
    - HEA funded project FETLAR
      - Migration of locked-in content from closed-format/source systems to QTI
      - Creation of the FETLAR Virtual Appliance - a pre-configured, easily deployable package including all the FETLAR content plus the QTI tools required to deliver it
  - Partner institutions past and present include Oxford, Cambridge, Glasgow, Edinburgh, Birmingham, Southampton, Harper Adams, Strathclyde and many more
Other Technology and Software:

- Virtual Lab Environments
  - VLab
    - Delivers a full, virtual computer environment to a remote web browser
    - Allows distance learning students to undertake a practical, computer-based workshop from home without having to configure their local machine
    - Bypasses any limitations of university lab equipment
  - Wlab
    - Adds the ability to create "staged" exercises, with a virtual machine representing each component of a practical workshop
For more detail:

- ltrg.kingston.ac.uk
- uniqurate.kingston.ac.uk
- aqurate.kingston.ac.uk
- www.paulneve.com/wlab
- paul@kingston.ac.uk
- graham@kingston.ac.uk
References

- Wulf, T. (2005), Constructivist approaches for teaching computer programming, In Proceedings of the 6th conference on Information technology education, SIGITE '05. ACM, New Jersey, USA