COMPUTING AT SCHOOL

EDUCATE · ENGAGE · ENCOURAGE

Computer Programming in Key Stage 3

September 2009

Editor:

John Woollard, School of Education, University of Southampton

Contributors:

Liz Crane, Head of ICT, Oaklands Roman Catholic School, Waterlooville, Hampshire

Roger Davies, Director of ICT, Queen Elizabeth School, Kirkby Lonsdale, Cumbria

Claire Johnson, Head of ICT, Westgate School, Winchester, Hampshire

David Kinsella, Head of ICT, The Nelson Thomlinson School, Wigton, Cumbria

Dan Marshall, ICT and computing teacher, Crofton School, Hampshire

Emma Wright, Head of ICT & Computer Science, Harvey Grammar School, Folkestone, Kent

Other contributions made from the CAS working group:

Roger Broadie, Roger Boyle, Luke Church, Paul Curzon, Roger Davies
Nick Efford, Mandy Honeyman, Simon Humphreys, Michael Kölling,
Jack Lang, Thomas Ng, Simon Peyton-Jones, Aaron Sloman,
John Woollard and Emma Wright.

Please address all queries to CPinKS3@computingatschool.org.uk

An electronic version is available at http://www.computingatschool.org.uk/files/CPinKS3/CPinKS3.pdf

Introduction

This document illustrates how the yearly objectives from the Framework for teaching ICT capability: Years 7, 8 and 9 can be grouped together and taught in a way that promotes and utilises knowledge and understanding of computing. Programming is a core activity of computing because it enables the user to access and release the potential of the computer they are using. Computer programming can be likened to playing chess - although there is a relatively small set of simple rules, it is the strategic and sustained application of those rules that can create interesting games between children or intellectual fights between grand masters. The same with programming, the first applications of the rules can produce the interesting results, fun play on graphics, numbers or words. But, there is no boundary preventing the learner moving all the way to being the grand master of computer programs. Once you can do it, the sky's the limit over what you can make computers do.

The following texts are direct quotes from the 2008 revision of the National Curriculum for ICT and they relate directly to programming activities.

> *Capability - using a range of ICT tools in a purposeful way to tackle questions, solve problems and create ideas and solutions of value.*
>
> *Developing ideas - pupils should be able to test predictions and discover patterns and relationships, exploring, evaluating and developing models by changing their rules and values.*
>
> *Use ICT to make things happen by planning, testing and modifying a sequence of instructions, recognising where a group of instructions needs repeating, and automating frequently used processes by constructing efficient procedures that are fit for purpose.*
>
> *Pupils should be able to review, modify and evaluate work as it progresses, reflecting critically and using feedback.*
>
> *Scope the information flow: Represent a system and identify all its parts, including inputs, outputs and the processes used. (Processes could include manipulating data or information.)*
>
> *Developing an ICT-based model to meet particular needs: this should involve testing predictions and discovering relationships, exploring, evaluating and developing models by changing their rules and values.*

The aim of the group ***Computing at School*** (http://www.computingatschool.org.uk) and this document is to promote the principles of "computing" into the key stage 3 curriculum. This document also celebrates, through using them as example scenarios, the work of teachers currently engaged in teaching programming to pupils. Particular thanks go to the staff and pupils of Harvey Grammar School, The Nelson Thomlinson School, Oaklands (RC) School, Queen Elizabeth School and Westgate School.

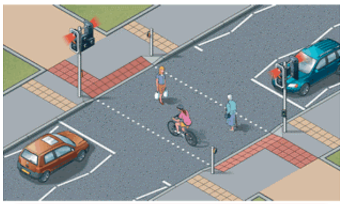
Contents

Please address all queries to CPinKS3@computingatschool.org.uk

Rationale for computer programming in the key stage 3 curriculum

In this basic example, pupils are introduced to the concept of sequencing instructions that will be followed by the computer to control the lights at a pedestrian/cyclist crossing (Toucan).

```
IF INPUT 1 ON THEN
SWITCH OFF 3 [traffic green light]
SWITCH ON 2  [traffic amber light]
WAIT 3
SWITCH OFF 2 [traffic amber light]
SWITCH ON 1  [traffic red light]
WAIT 2
SWITCH OFF 4 [pedestrian/cyclist red light]
SWITCH ON 5  [pedestrian/cyclist green light]
WAIT 20
SWITCH OFF 5 [pedestrian/cyclist green light]
SWITCH ON  4 [pedestrian/cyclist red light]
WAIT 5
SWITCH ON 2  [traffic amber light]
WAIT 5
SWITCH OFF 1 [traffic red light]
SWITCH OFF 2 [traffic amber light]
SWITCH ON 3  [traffic green light]
ENDIF
```



Rule 25: Toucan crossings can be used by both cyclists and pedestrians

http://www.direct.gov.uk/en/TravelAndTransport/Highwaycode

This type of program is just a plan-of-action a machine can follow and everything the computer does is based upon the plan-of-action. There are different ways in which computer programming can be approached, there are different forms that computer programs can take and there are different resources/software available to create programs in particular forms.

The program (procedure) above is called whenever the button (INPUT 1) is pressed. Whenever, a pedestrian walks up to the crossing and presses the button. Even a simple sequence of events such as this can introduce the pupils to the vocabulary of programming such as:

sequence

algorithm

input

output

command

operator operand

condition / decision

repetition (iteration)

The first example is an imperative, high level, third generation approach to programming (common in schools in the form of Flowol, LOGO, BASIC, etc.) The alternative in commercial computing is object orientated programming (OOP) exemplified by JAVA, Greenfoot, Scratch, etc. and declarative languages, exemplified in schools by Prolog.

Why teach programming?

We believe that teaching programming is important for two core reasons:

firstly, it is a form of digital literacy that is of growing importance within society; and

secondly, it promotes intellectual development and the development of problem-solving skills in a way that is applicable to many other subjects and in many other areas of life.

The first point relates closely to the Every Child Matters agenda and the core principles of “enjoy and achieve”, “make a positive contribution” and “achieve economic well-being”. http://www.dcsf.gov.uk/everychildmatters/about/aims/aims

The second point relates to the current initiatives in PLTS (Personal Learning and Thinking Skills) http://curriculum.qcda.gov.uk/key-stages-3-and-4/skills/plts/index.aspx

Sample teaching unit – computer programming

Programming as digital literacy

Computers are now instrumental to our society and the need for pupils to attain a form of ‘digital literacy’ is now generally accepted. This is currently interpreted as the need to be able to use standard applications, such as office-type software within a windows environment interface, proficiently. We agree that this is important.

However, the use of computers is changing rapidly. They are now as much mechanisms for social communication, as they are office tools. As this connectivity expands to every aspect of our lives, the ability to exercise control over the information becomes crucial. Controlling information is one of the fundamental skills of programming. If students master this skill, they will be able to engage successfully, not just with today’s applications, but also with uses of technology that have yet to be devised. Academic support for this comes from Church and Whitten (2009) and Blackwell’s Attention investment (1999).

Programming offers the ability to create new uses for computers. Whereas a competence in office-type software allows the production of new documents, programming allows the creation of new behaviours, rather than just the consumption of behaviours provided for us by others. Wing (2006) argues really what is involved is the act of ‘Computational Thinking’, which is fundamental to many branches of both art and science.

Computer programming is carried out by many people and is a hobby, pastime, leisure pursuit, interest, diversion, relaxation… For our pupils, it could be a way of enabling them to “enjoy and achieve” - an aim of the Every Child Matters agenda http://www.everychildmatters.gov.uk/aims

"*In some senses, computer programming itself is one of the best computer games of all. In the ‘computer programming game’, there are obvious goals and it’s easy to generate more. The ‘player’ gets frequent performance feedback (that is, in fact, often tantalizingly misleading about the nearness of the goal). The game can be played at many different difficulty levels, and there are many levels of goals available, both in terms of the finished product (whether it works, how fast it works, how much space it requires, etc.) and in terms of the process of reaching it (how long it takes to program, etc.). Self-esteem is crucially involved in the game, and there is probably the occasional emotional or fantasy aspects involved in controlling so completely, yet often so ineffectively, the behaviour of this responsive entity. Finally the process of debugging a program is perhaps unmatched in its ability to raise expectations about how the program will work, only to have the expectations surprisingly disappointed in ways that reveal the true underlying structure of the program*" (Malone, 1980).

Computer programming is also a vocational pursuit and may enable our pupils to “*achieve economic well-being*”, another aim of Every Child Matters. Pupils discovering their proficiency in handling syntax, algorithm, logic and analysis may find they can enter an industry in which those skills are highly valued.

This teaching resource is designed to bring computer programming to every pupils’ experience because it contributes in a powerful way to their ability to learn, conceptualise and understand. Computer programming is a subset of a computing curriculum and, together with other knowledge, experience, skills and understanding of computing, it exercises skills that are valuable in other aspects of learning, work and leisure. It also gives an insight into why computers behave as they do and therefore puts an understanding into the ICT curriculum.

Hence, programming:

- enables pupils to enjoy and achieve;
- develops problem-solving skills through both individual endeavour and team work;
- provides experience of a powerful way to "learn, conceptualise, and understand".

The next section identifies six aspects of programming that bring wider benefits to the way pupils think and learn.

Commentary

“I'd add that programming is a tremendous *motivator* for students, because it makes computers ‘come alive’ and ‘dance to their tune’. Maybe the language of "computational thinking" belongs here? One way to put it is this is ‘programming makes abstraction concrete’. ‘Abstraction’ is a tremendously powerful idea that we use again and again but it is, by definition, abstract. Programming gives tangible, concrete form to the act of abstraction, and repeatedly shows how useful it is. You don't have to say ‘we're going to learn about abstraction’. You just do it, repeatedly, and then afterwards say ‘look at the common pattern that we have used over and over’” Simon Peyton-Jones. Teaching Programming for Cognitive Benefits

Teaching this key skill of computational thinking, through teaching programming, allows pupils to develop a new cognitive tool.

Aspects of programming

This analysis of the pedagogic value of programming identifies 6 important areas: accuracy of expression, understanding algorithms, visual representation of concepts and procedures, analysis of situations, data structures and application of logic.

Accuracy of expression

Through computer programming we can insist upon and, importantly, demonstrate the need for accuracy and precision in what we do. However, this does not prevent creativity. Like a chess player has to obey the rules of the game, the imaginative and thoughtful implementation of rules can create structures and procedures that are unique and valued.

At the ***character level*** it is akin to spelling in the English curriculum - color and colour are significantly different.

To change the colour of text to red use `<font color="#FF0000">` [HTML]

But the accuracy at character level is more than just an Americanisation of spelling; it is also “paying attention to detail” and realising that spaces, punctuation marks and the case of letters are important. In many areas, computer programming is case sensitive.

At the ***syntax level*** the pupils become aware of the structures of instructions with operands and operators and the need to match each with the other in much the same way as there is an object-verb relationship. For example, `FORWARD 10, WAIT 20, REPEAT 5, PRINT “hello world”.`

Structures such as `IF THEN ELSE ENDIF` emphasise the importance of accuracy at a level higher than the individual character. In science, pupils would be expected to know the symbols of reactants in an experiment and use the correct syntax when representing them, for example,
$\mathbf{H_2SO_4 + 2NaHCO_3 \rightarrow Na_2SO_4 + 2H_2O + 2CO_2\uparrow}$

At the ***instruction level*** the pupils have to be aware of structure in the same way as the grammar of an English sentence has rules of structure. All sentences begin with a capital letter and end with a full stop, exclamation or question mark. They follow the rules of grammar and punctuation. Each sentence has a subject and a verb. In the same way, computer instructions have a precise structure with common rules called syntax. For example, in many languages the end of a line of code is indicated by a semi-colon.

```
    if (aName == bName) {System.out.println('== the same')};
```

At the ***module, procedure or program level*** the concept of wholeness is introduced. The computer program is complete within itself and usable. When pupils are asked in history to write about the reason for the rise of nationalism in the 1930s, they are expected to present their ideas as a sequence of connected statements that follow each other logically to build a sustained argument, frequently as a paragraph of text. In the same way, pupils develop the skills of coherent thought through sequencing the instructions, beginning by setting the context, carrying out the operation and concluding in a formal manner. Writing a complete module, procedure or program is like writing a formal essay, poster, recipe or invitation. The product has a wholesomeness or completeness.

Two examples, the first is a simple program in BASIC to print out a “times table”,

```
    10 A=7
    20 M=12
    30 FOR K = 1 TO M
    40 P=K*A
    50 PRINT A;” times “; K;” equals “; P
    60 NEXT K
    70 END
```

Lines 10 and 20 set the context; 30 to 60 carry out the repeated operation to print 1 times 7 equals 7, 2 times 7 equals 14, etc. and the final line formally ends the program releasing the computer to do other things.

The second example completes the same task in Pascal

```
    a := 7;
    m :=12;
    for k := 1 to m do begin
      p := p * a;
      writeln(a, " times ", k, " equals ", p);
    end;
```

Sample teaching unit – computer programming

Commentary

“The rigorous nature of programming and the need for absolute precision seem to surprise a fair few of our [undergraduate] students when they encounter programming in the first few weeks of their Computing/CS degrees, and some of them clearly struggle to come to terms with it. Perhaps this need for discipline and precision is something we should be promoting more strongly and at a much earlier stage in their education - but do we then risk stifling the creative element and making programming seem that much more dull and difficult?” Nick Efford

“For many years there have been interactive programming development environments (including several developed by Artificial Intelligence researchers, as well as others) that support incremental development and testing, and are perfectly capable of giving sensible feedback if the programmer makes a trivial mistake, for example, and allowing the programmer (e.g. a learner) to find out what went wrong, correct it and continue -- without having to start all over again. Of course, deep mistakes, where your program runs, but does the wrong thing, are another matter. Experiencing that is an important form of education. Having devices that prevent all such errors is the last thing we need in our educational environments.” Aaron Sloman

“To me programming at that stage [key stage 2] is a way of exploring the virtual world. It’s practical mathematics. Theory is fine, but it needs a base of experience to build on and be relevant. I still remember the thrill of writing a program that actually (eventually) did something, even if only to print Hello World.” Jack Lang

Understanding algorithms

Algorithm can be considered a sequence of instructions, a finite set of commands or a method of working. Algorithm can be considered synonymous with program but algorithm encompasses the whole domain of carrying out instructions in a predefined and accurate way. Algorithm is to computer program as writing is to story. It is not limited to programming. However, through computer programming, pupils can gain a better understanding of the value of predefined sequences of action to more efficiently and effectively achieve an outcome.

Two initial definitions for pupils are:

an algorithm can be represented as a sequence of instructions to be carried out until an end point is reached;

algorithms are the rules, conditions or sequence by which the computer or people tackle a problem or situation.

Other keywords to be used when discussing algorithm are:

steps, instructions, commands

sequence, flow,

decisions, branches, jumps, if then, conditional, if then else, true false

repeat, until, condition, iteration

Algorithm takes many forms. It can be the rules by which you drive a car. It can be the way in which you eat from a buffet. It can be the way in which you carry out a science procedure.

“You are approaching a traffic queue: which lane do you take? Always going to the shortest line is a greedy algorithm. You might consider the shortest queue but always err to the right because you think that the fastest drivers are there. That algorithm is more strategic or refined. You may rely upon local knowledge of the road and queues and make different decisions in different places. The algorithm is very specialised and contains or makes reference to information. In a similar way, we program the computers to obey a set of predetermined rules - the algorithm.” John Woollard

Analysis of situations

The act of creating a computer program usually requires a deeper and more rigorous analysis of the context of the program than a pupil might otherwise undertake, for example, the sequence of traffic lights or the input/output requirements of a heating system or the data requirements of a video shop. The use of techniques such as systems analysis gives pupils an insight into the ways in which precision can be obtained. The analysis of system diagrams help pupils to understand how complex systems work. Those systems may be anything from biological population, mechanical devices, businesses, through to the impact of social policy and regulations. The synthesis of system diagrams for familiar situations helps the pupils understand the detail and the complexity of systems – for instance, the school as a system. At the highest level (simplest) view of a system it is similar to a block diagram, showing its inputs, outputs and processes. System diagrams and data flow are particularly helpful in showing how a change in one factor may impact elsewhere. Importantly, a good diagram might show how changing a factor may feed back to affect itself!

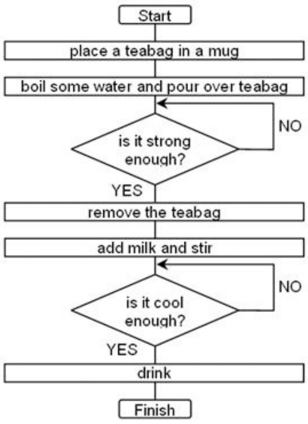
Drawing a system or data flow diagram is a good way of starting to build a computer model. The technique helps you to map out the structure of the system to be modelled. It shows the factors and relationships that are important, and helps you to start quantifying the linkages between factors.

Visual representation of concepts and procedures

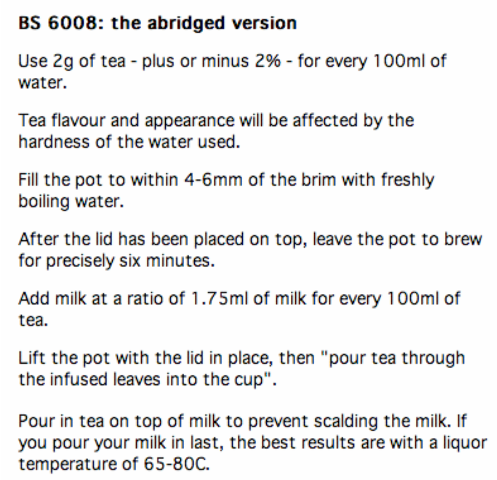
The skills of visual representation of pupils' understanding are acknowledged in many areas of the curriculum. In English there is the story board, in mathematics is the chart, in science the symbolic representation of atomic and subatomic particles, in geography is the map at different levels of scale and symbolism and in humanities are the icons of religion, consumerism and politics. In computing there are visual representations of what the computer is doing when running a program: flow diagram, variables table and structured English instructions. Each aids the computer programmer but also, each is a tool that learners can apply in other areas of study to help them more efficiently and effectively represent their knowledge and understanding.

The following illustrates different representations of "making a cup of tea" – the structured English versions reflect the rigour and precision required when preparing to program a computer.

A picture tells a thousand words



A flow diagram



BS 6008: the abridged version

Use 2g of tea - plus or minus 2% - for every 100ml of water.

Tea flavour and appearance will be affected by the hardness of the water used.

Fill the pot to within 4-6mm of the brim with freshly boiling water.

After the lid has been placed on top, leave the pot to brew for precisely six minutes.

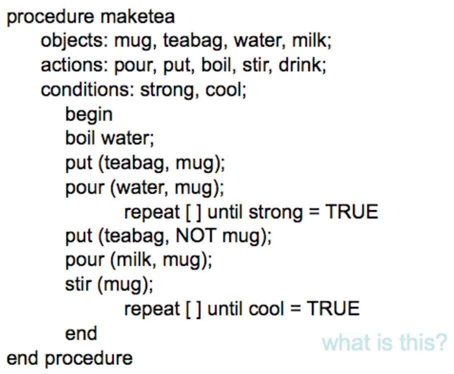
Add milk at a ratio of 1.75ml of milk for every 100ml of tea.

Lift the pot with the lid in place, then "pour tea through the infused leaves into the cup".

Pour in tea on top of milk to prevent scalding the milk. If you pour your milk in last, the best results are with a liquor temperature of 65-80C.

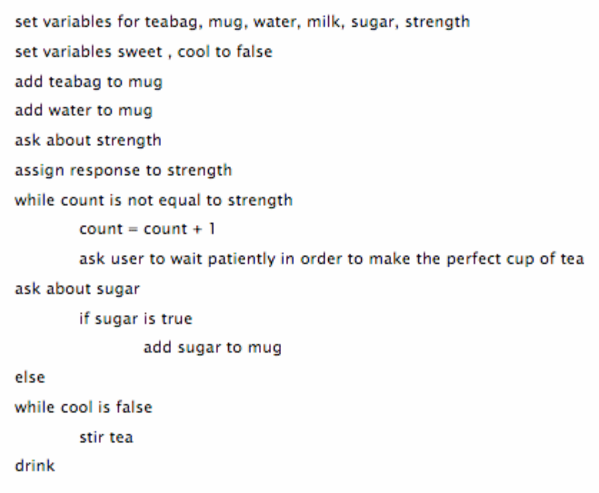
http://www.guardian.co.uk/Archive/Article/0,4273,3908389,00.html

Pedantic English



```
procedure maketea
    objects: mug, teabag, water, milk;
    actions: pour, put, boil, stir, drink;
    conditions: strong, cool;
        begin
        boil water;
        put (teabag, mug);
        pour (water, mug);
             repeat [ ] until strong = TRUE
        put (teabag, NOT mug);
        pour (milk, mug);
        stir (mug);
             repeat [ ] until cool = TRUE
        end
end procedure
```

what is this?



```
set variables for teabag, mug, water, milk, sugar, strength
set variables sweet , cool to false
add teabag to mug
add water to mug
ask about strength
assign response to strength
while count is not equal to strength
        count = count + 1
        ask user to wait patiently in order to make the perfect cup of tea
ask about sugar
        if sugar is true
                add sugar to mug
else
while cool is false
        stir tea
drink
```

Structured English

Data structures

In computing, a data structure is a particular way of storing and organising information so that it can be accessed efficiently. Different kinds of data structures are suited to different kinds of applications and different types of data. Three examples that pupils will be familiar with are: classification keys, indexes and labels. In science, classification keys in biology or chemicals are often based on a tree structure. Paper-based shopping catalogues use indexing and sequencing of the information. Car registration number plates represent the systematic labelling of items. Pupils can learn to interpret (decode) a number plate and begin to understand the process of labelling for computerised systems such as the bar codes on goods for sale, the structure of URLs or the unique identifiers such as National Insurance numbers and the use of check digits.

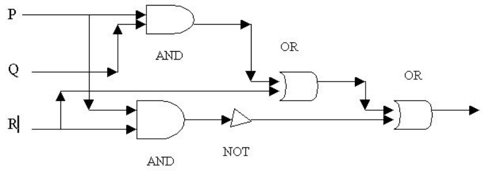
Some data structures are specifically designed for efficient computer processing, for example, B-trees are particularly well-suited for implementation of databases, while compiler implementations usually use hash tables to look up identifiers. Data structures are used in almost every program or software system. Specific data structures are essential ingredients of many efficient algorithms, and make possible the management of huge amounts of data, such as large databases and internet indexing services. Some formal design methods and programming languages emphasize data structures, rather than algorithms, as the key organizing factor in software design.

Application of logic

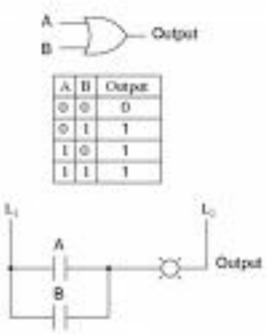
Logic as a topic encompasses the understanding and application of the Boolean algebra AND, OR, NOT truth values in keyword searching, electronic circuits, logic circuits and truth tables. This rigorous application of rules again supports the need for accuracy of expression and clear thinking around novel concepts. Logic is the application of methods and criteria of validity of inference, reasoning and knowledge. The following are examples of the application of logic.

Boolean logic is used in search engines to retrieve items on the web, for example, when searching in Google, if you enter words adjacent to each other, say monty python, the interface automatically inserts an AND operator between the words and returns documents or items which contain both of these words, not necessarily adjacent to each other. The search for monty OR python would give many more hits whereas the search for “monty python” would give much fewer hits.

Logic is used in the construction of electronic circuits and represented in truth tables.



Dan Buzan, University of Boston http://cs-people.bu.edu



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

An important concept in the computing curriculum is “logical device” in which devices are names logically (A, B, C…) and not by their physical name.

Comment

As an alternative analysis, Roger Broadie observes, “I believe ICT **[computing]** has a unique contribution to offer in that it brings ways of thinking that are not provided for or by any other areas of the curriculum… These ways of thinking certainly include (and there may be others):

- Programming; analysis of processes in order to produce complete and correct programs that will provide the desired result. This is an important skill whether or not ICT is involved.
- Interface design; this is about human interactions with information and other humans, and the ways in which information and the development of the interaction are presented to stimulate and guide the interaction. Other subjects touch on this but none with the depth and effectiveness that ICT can.
- Information structuring; this includes hierarchical, relational and hyper-linked structures, and while some of these are covered in say science with biological keys, relational and hyper info structures can only be satisfactorily worked on with ICT.
- Networked communication; even simple examples such as how to use the asynchronous nature of email effectively are hardly covered in the English subject curriculum, and where social networking is taking us most certainly is not.
- Language structure and semantics; while this is shared with human language studies when they look at grammar, there is a broad range of semantic structures used in programming languages that human language does not use.
- Data; including coding of data, data redundancy and issues around compression of data.
- Search and information validation.

While some of these might be beyond the school curriculum except for the brightest who specialise in ICT, I would maintain that the first four at least should be studied to some level by all, as they are vital to how life and work will operate this century. And ICT as a separate subject is the way to do this.”

ICT sample teaching units – some scenarios

The way in which computer programming can be introduced in the classroom is illustrated through scenarios. These scenarios are not comprehensive in nature but illustrative of ***good and successful*** practice in UK schools. Each scenario is described by the opportunities to support particular forms of computer programming. The outcomes of the pupils' activities are described in terms of the attainment target levels. Alternative resources are also described.

The following sections are adopted directly from a sample teaching unit. "*The ICT Framework recommends that schools offer one hour each week, or 38 hours per year, for discrete ICT lessons. The sample teaching units for a year, if taught without amendment, need less teaching time than 38 hours. This leaves time for lessons of your own design at suitable points*" (DfES, 2002a). Now updated: http://nationalstrategies.standards.dcsf.gov.uk/secondary/secondaryframeworks/ictframework

There is no requirement to use the DfES units but they have been adopted and adapted in many schools. They normally contain sample lesson plans that you can amend to suit your local circumstances and the needs of your pupils. This unit, Computer Programming in Key Stage 3, is different in that it presents alternative scenarios that can be taught with one of several different resources.

The units contain outline plans for lessons of 60 minutes although the nature of the pupils, their prior experience, their aptitude to take on new ideas and the resources available will determine the rate by which pupils can progress through the activities.

The scenarios introduce some of the ICT Framework objectives for Year 7 in the theme 'Developing ideas and making things happen'. The scenarios focus upon the National Curriculum (QCA, 2007) key processes of developing ideas, communicating information and evaluating. In particular, the activities support curriculum requirements that pupils should be able to:

2.2e use ICT to make things happen by planning, testing and modifying a sequence of instructions, recognising where a group of instructions needs repeating, and automating frequently used processes by constructing efficient procedures that are fit for purpose;

2.3c use technical terms appropriately and correctly;

2.4a review, modify and evaluate work as it progresses, reflecting critically and using feedback.

These statements are taken from the "new" National Curriculum introduced in September 2008 to Year 7 pupils.

"Reflecting critically could include self-review, peer evaluation and user or audience feedback. Pupils should judge both the quality of their work and how effectively they have used ICT." In computer programming this can be reflected in the minimum use of code, the fastest processing time or the shortest development time.

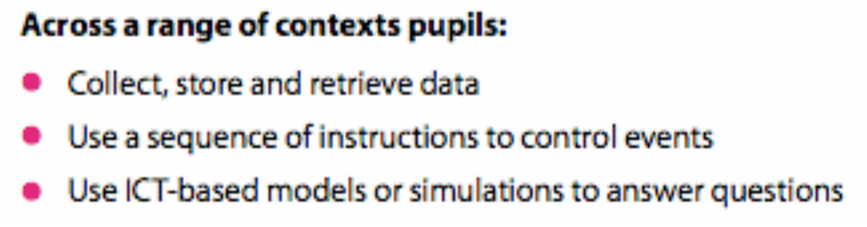
"*They use appropriate evaluation criteria to critically evaluate the fitness for purpose of their work as it progresses.*" The curriculum offers the opportunity for the pupils to learn about efficiency of coding and algorithm and the need to add remarks/comments to aid future development.

Aspects of control and monitoring are taught in both science and design and technology. You might find it helpful to ask these departments what they have covered with pupils before you teach this unit. You could then refer to the work pupils have done in these other subjects at appropriate points in the lessons. For example, pupils may have created sequences of instructions in control software. The following statements are drawn from the DCSF publication Assessing pupils' progress in ICT at Key Stage 3 (DCSF, 2008). They illustrate AF2 - Handling data, sequencing instructions and modelling at the different levels of attainment.

department for
children, schools and families

01012-2008POS-EN

Level 3

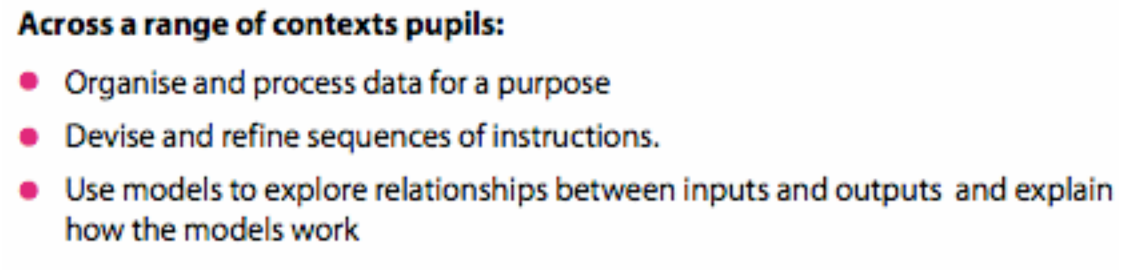


Across a range of contexts pupils:

- Collect, store and retrieve data
- Use a sequence of instructions to control events
- Use ICT-based models or simulations to answer questions

Sample teaching unit – computer programming

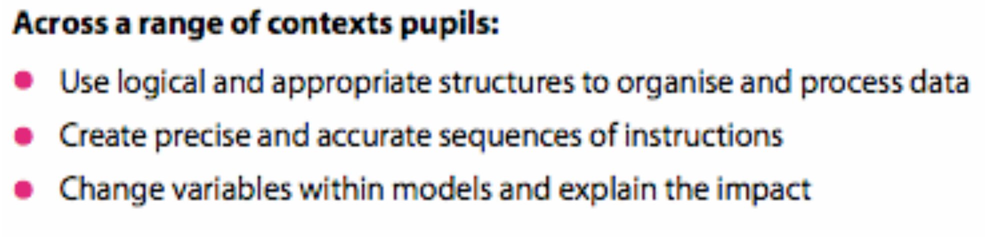
Level 4



Across a range of contexts pupils:

- Organise and process data for a purpose
- Devise and refine sequences of instructions.
- Use models to explore relationships between inputs and outputs and explain how the models work

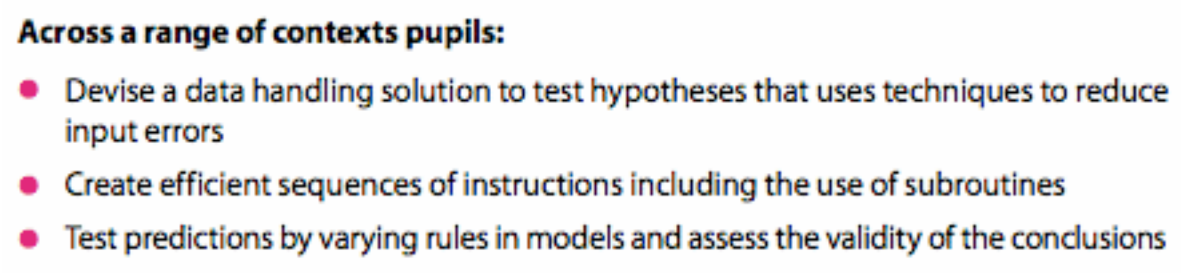
Level 5



Across a range of contexts pupils:

- Use logical and appropriate structures to organise and process data
- Create precise and accurate sequences of instructions
- Change variables within models and explain the impact

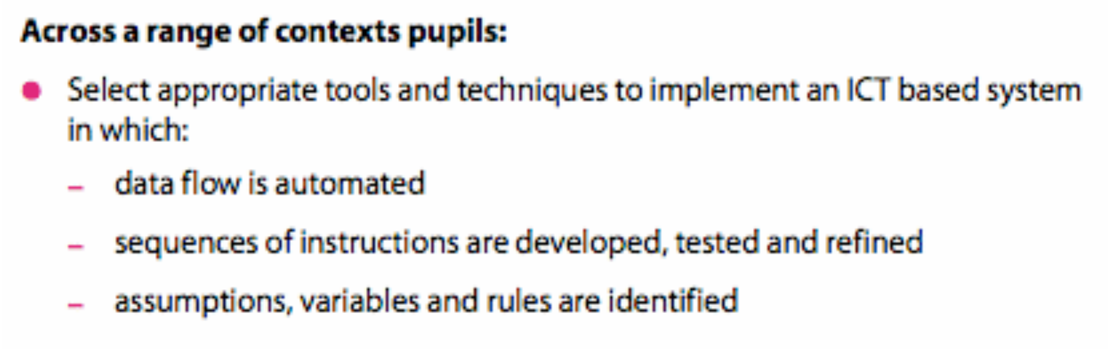
Level 6



Across a range of contexts pupils:

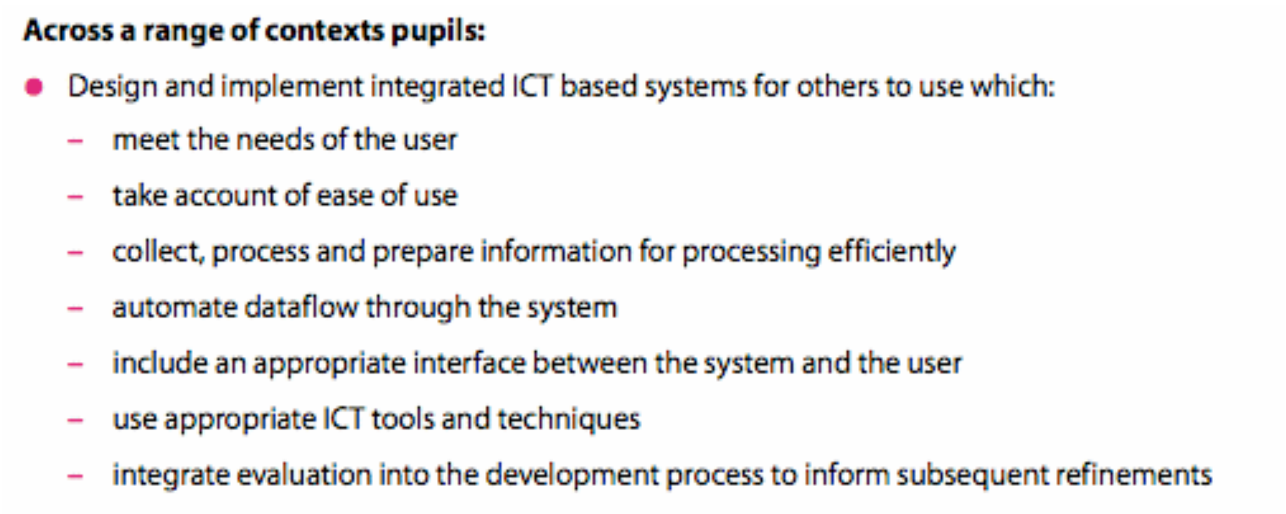
- Devise a data handling solution to test hypotheses that uses techniques to reduce input errors
- Create efficient sequences of instructions including the use of subroutines
- Test predictions by varying rules in models and assess the validity of the conclusions

Level 7 and Level 8



Across a range of contexts pupils:

- Select appropriate tools and techniques to implement an ICT based system in which:
  - data flow is automated
  - sequences of instructions are developed, tested and refined
  - assumptions, variables and rules are identified



Across a range of contexts pupils:

- Design and implement integrated ICT based systems for others to use which:
  - meet the needs of the user
  - take account of ease of use
  - collect, process and prepare information for processing efficiently
  - automate dataflow through the system
  - include an appropriate interface between the system and the user
  - use appropriate ICT tools and techniques
  - integrate evaluation into the development process to inform subsequent refinements

Scenario 1 Computer Programming using Alice

This unit is designed to introduce programming to pupils through the open source programming package Alice (http://www.alice.org). Alice contains a library of images and backgrounds for pupils to use. The tutorials referred to in this teaching unit are also accessible from within the software.

The unit is designed and used by Emma Wright, Head of ICT & Computer Science, Harvey Grammar School, Folkestone, Kent.

Alice is a tool that is designed to introduce programming concepts without the need to launch directly into learning code. This project focuses upon the use and creation of events and methods that enable the user to create their own virtual world, complete with animated characters. A feature of this project is that it also introduces the pupil to problem solving techniques that will help them to progress their skills by breaking down problems using Hierarchy Charts, Stepwise Refinement and a State Transition Diagrams (see glossary).

This project covers 7 lessons, which are approximately 50-60 minutes in duration, and are structured as:

1 Introduction to Alice & vocabulary; Inputs, Processes & Outputs

2 Creation of new methods; Hierarchy Charts

3 Events; Stepwise refinement

4/5/6 Project: Design, Development, Evaluation

7 State transition Diagrams Presentation of work for display.

The activities and curriculum content are mapped against aspects of the National Curriculum for ICT (2007).

National Curriculum mapping

1.1a ✓

1.1b ✓

1.1c ✓

1.2a

1.3a ✓

1.3b

1.3c

1.4a ✓

1.4b

1.5a

1.5b ✓

2.1a ✓

2.1b

2.1c

2.1d ✓

2.2a ✓

2.2b ✓

2.2c ✓

2.2d ✓

2.2e

2.2f ✓

2.3a ✓

2.3b

2.3c ✓

2.4a

2.4b ✓

2.4c

3a

3b

3c

3d

3e

4a

4b

4c ✓

4d

4e

4f

The Alice environment resources for this module are in the Gallery of the installed copy of Alice and should be directly accessible. An online gallery is available at: http://www.alice.org/gallery

Other resources for this module are available at

http://computingatschool.org.uk/files/CPinKS3/Alice

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
1/7	Know that a computer program is a series of instructions programmed in to automate a sequence to achieve a useful result. Know that an object can perform a process or a series of steps and that methods can be applied to an object and that a computer can change an array of inputs, process them into something new. Understand that a process consists of working with a range of inputs which when processed create a given output. How to apply methods to an object within a programming environment.	Assessed deliverables: completion of Tutorial 4 completion of Tutorial 1 completion of IPO diagram.	Importance of computer programming to make computers do what we wish them to do.	In this order: Gallery - Tutorial 4 – Beach House Gallery - Tutorial 1 – Ice Skater Key words include: Method Object Detail Event Editor Property Input Process Output	1.1a 1.1b 2.2a 2.3c
	Starter: How do you make a cup of coffee? What are the inputs, process, and output? Talk: ALICE environment. Describe that the software enables animations to work. Hands on: Understand the concepts of a 'World' and 'actors' within that world. Introduction – tutorial 4. Hands on: Introduction to Alice. Introduce the dancer as an 'Object'. Pupils to run through the first tutorial: Change the dancer's sequence. Talk: What are the inputs/process/outputs? Present this in diagrammatic form using a whiteboard. Use some examples. Activity: Pupils to complete this process as an activity for the movement of the dancer. Talk: Explain that everything we do on a computer needs a purpose. An end result. Extension: To experiment moving actors in the library. Plenary: Oral quiz of the keywords. Homework: Download and install the ALICE software or gain access to Alice in out-of-school/study time.				

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
2/7	Know that new ‘methods’ are being created all the time in the world – and that Computer Science is responsible for some of them! Understand how to create a new method and what these methods can be. Relate movement back to the overall purpose of the animation. Know that a hierarchy chart can be used to represent a numbered list of steps. Thus enabling problems to be broken down into logical steps.	To create a new method using ALICE as a tool and to create a hierarchy chart to represent the decisions you took to create this new method. Understand that a problem can be broken down into smaller ones and that a hierarchy chart can be used to diagrammatically represent steps. Assessed deliverables: Tutorial 2 Creation of a hierarchy chart.	Programming enables boundaries to be broken and new things to be created.	Gallery - Tutorial 2 – Defending Nap time Hierarchy Chart example	1.1a 1.1b 2.2a
	Starter: Recap: Quiz on keywords of last lesson. Discussion: Are new objects and new methods being created all the time in the real word? Discuss advances in technology. E.g. iPod generations. Hands on: ALICE: Creation of a new method ‘to make the bunny jump’. Discussion: How do we diagrammatically represent solutions to problems? (The Hierarchy Chart). Hands on: Pupils to create their own hierarchy chart to the bunny jump. Discussion: Does breaking down a problem into sub problems help make them easier to solve? Homework: Think of new method that can be applied to a technological gadget of your choice. Explain what the new method will do and how the user might benefit.				

<table>
<tr><th>Lesson Number</th><th>Learning objective
We are learning to
WALT</th><th>Learning outcomes
What I am looking for
WILF</th><th>Rationale
This is because
TIB</th><th>Prerequisite skills of pupil and teacher and resources</th><th>NC, Strategy and QCA references</th></tr>
<tr><td>3/7</td><td>That programming enables events to be created that can be controlled by the user.
Those events comprise of logical steps that can be broken down.</td><td>That objects can be manipulated by the user in the form of a key press or mouse click.
That the process of stepwise refinement is used to create algorithm, which is the description of a process that achieves some task.
Assessed deliverables:
Tutorial 3
Stepwise Refinement task.</td><td>Programming enables us to control what we can do on screen – something we can all take for granted.</td><td>Headphones
Gallery – Tutorial 3 –
Penguin Song
Stepwise refinement example.</td><td>1.1a
1.1b
2.2a</td></tr>
<tr><td></td><td colspan="5">Starter: Discussion about homework. Pupils are encouraged to talk about their ideas.
Hands on: ‘Go through tutorial ‘Penguin Song’.
Discussion: Pupils to think about the steps that they went through to create the effect.
Talk about Stepwise refinement. Show an example to explain and discuss. E.g. Making a burger. Toast bun, grill burger, add toppings etc.
Hands on: Students using MS Word to write a stepwise refinement plan for their Penguin Song. (May require help and guidance from teacher).
Plenary: When we use Office software, what are the ‘events’ that we produce and what controls are at our disposal?
Homework: Use stepwise refinement to create a set of instructions to travel from Birmingham to Marseille.</td></tr>
</table>

Sample teaching unit – computer programming

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
4/7 5/7 6/7	That objects can move via pre-existing methods and newly created ones. The importance of designing a system prior to its development. The importance of user requirements. Know that events need to relate back to the overall purpose of the use of the model.	Be imaginative and develop a virtual world that is to then be used by another user for a predefined purpose. Know how to document a project. Assessed deliverables: Design Technical Evidence Evaluation Homework tasks	All programs developed need to have a useful purpose and a demand. That commercial programs have a value.	Writing frame for the design of the project. Writing frame for the evaluation. Example projects for the less and more able.	1.1a 1.1c 1.3a 1.5b 2.1a 2.1d 2.2a 2.2b 2.2c 2.2f 4c 2.4b
	Starter: What do you think you could create using Alice – in two lessons that would a) have a purpose and b) be useful? One example is to have a keyboard that could teach the user what each note sounds like. Discussion: Of ideas generated, write best ideas on board to share with the class. Hands on: Write up plan with a simple writing frame. *Pupils can work in groups should they wish. Hands on: Pupils to then start to produce their own virtual world making sure that they work from their own plan. When completed: Pupils then test out each others virtual world on the basis of a) fit for purpose, and b) usefulness to other users – I.e. would there be a demand for the product? The winner is the group/student with the best peer-assessed mark. Plenary: What projects did you like? What made you like them? Draw upon good design principles. Homework: (use previous examples from lessons 2 & 3 to help). Week 4: Create a hierarchy chart for your project. Week 5: Use stepwise refinement to show how your project can be broken down into a series of smaller steps. Week 6: Ensure all project work is completed by either using the software at home or using the facilities in school.				

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
7/7	Introduction to Finite state automata Presentation of work for display. That work of a computer scientist is unique, and focuses upon the development of the system.	That processes change the state of inputs to produce an output. Presentation of work for display/e-portfolio/folders. Assessed deliverables: 12. State Transition table.	There are differences between ICT and Computing which need to be made apparent.	Resources: Display boards, backing paper, stapler etc. Example of a state transition diagram. Video	1.1a 2.3a 2.3c 1.4a
	Starter: Identify any particular processes that cause an/or many inputs to change state. Hands on: Teacher to discuss Finite State Automata and ask pupils to create one state transition diagram for a chosen event that can change state. Students to ensure they have compiled: Hierarchy Chart. Stepwise Refinement. State Transition Diagram. Annotated print screens of virtual world project & design/evaluation documentation. Any task that has not been completed or needs to be improved can be given extra opportunity during this lesson. Plenary: What is the difference between ICT & Computer Science? What have been the sole characteristics of using Alice that have made the two disciplines different? How does Computer Science affect our every day lives?				

Other useful resources o support programming in Key Stage 3 are available on the Computing at School website:

http://www.computingatschool.org.uk/files/CPinKS3/Alice

Scenario 2 Greenfoot and Game Design

This unit is designed to introduce programming to pupils who have experienced other pre-coding programming activities such as the visual interfaces of Scratch or Game Maker. It is based upon the open source programming application Greenfoot (http://www.greenfoot.org). Greenfoot combines the sophistication and power of coding in a mainstream, commercial programming language (Java) with activities based on a visual two-dimensional grid. The development environment uses class browser, editor, compiler, execution, etc. (see glossary) but is suitable for novice programmers.

The unit is designed and used by Emma Wright, Head of ICT & Computer Science, Harvey Grammar School, Folkestone, Kent.

This project covers a minimum of 7 lessons, which are approximately 50-60 minutes in duration, but the final "game authoring" task may be extended over a number of lessons.

1 Introduction to programming in Java

2 Moving objects, using a Mover class

3 Integrating image editing with programming

4 Interactivity – using the mouse click event

5 Considering game design

6 Logic and status

7 Creating a game

The activities and curriculum content can be mapped against many aspects of the National Curriculum for ICT (2007).

National Curriculum mapping

1.1a ✓
1.1b ✓
1.1c ✓
1.2a
1.3a ✓
1.3b

1.3c
1.4a ✓
1.4b
1.5a
1.5b ✓

2.1a ✓
2.1b
2.1c
2.1d ✓
2.2a ✓

2.2b ✓
2.2c ✓
2.2d ✓
2.2e
2.2f ✓

2.3a ✓
2.3b
2.3c ✓
2.4a
2.4b ✓
2.4c

3a
3b
3c
3d
3e

4a
4b ✓
4c ✓
4d
4e
4f

Please note:

The Greenfoot website has a large range of alternative approaches. In particular, the video resources are extremely useful for the independent learner. They can enable the more able pupils to progress independently.

http://www.greenfoot.org

The Introduction to Programming with Greenfoot by Michael Kölling is an entry-level introduction that systematically takes the learner through the basics of Object Oriented Programming in a clearly illustrated and pedagogically sound way and is suited to key stage 3 teaching.

http://astore.amazon.co.uk/pgce-21/detail/0136037534

Resources for this module are available at

http://computingatschool.org.uk/files/CPinKS3/Greenfoot.zip

They include: Tutorial1, Tutorial2 and debris.png.

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
1/7	That a computer program is a series of instructions designed to automate a sequence to achieve a useful result. That an object can perform a process or a series of steps and that methods can be applied to an object. Begin to understand the principles of object oriented (OO) programming. What is a high level language? Understanding that a process consists of working with a range of inputs which when processed create a given output.	How to apply methods to an object within a programming environment. Assessed deliverables: Changing backgrounds and adding actors.	Computer programming presents the user with the power to make the computer do what they want it to do.	In this order: Opening software Creating a blank scenario Changing the background Creating an actor Tutorial1 pp2-6 Key words include: Scenario World Actor Classes	1.1a 1.1b 2.2a 2.3a 2.3c
	Introduction to Java Programming - review: what is a high level language and discuss OO programming Introduction to Project - pupils will be expected to design their own game, using skills learned throughout the course. Show them the Asteroids game, as an indicator of what they could accomplish given hard work and dedication! Introduction to Greenfoot and introduce keywords. Pupils are to place a ladybug on their world that has a wall as a background, save for the next lesson. Talk: Explain that everything we do on a computer needs a purpose. An end result. Plenary: Oral quiz of the keywords.				

<table>
<tr><th>Lesson Number</th><th>Learning objective
We are learning to
WALT</th><th>Learning outcomes
What I am looking for
WILF</th><th>Rationale
This is because
TIB</th><th>Prerequisite skills of pupil and teacher and resources</th><th>NC, Strategy and QCA references</th></tr>
<tr><td>2/7</td><td>Know that a Java class is a group of Java methods and variables and a Java method is a set of Java statements that can be included inside a Java class.
Understand the development of code with a Mover class and inheriting methods from a class.
Use a code editor to add code to make actors move.</td><td>Level 5: They create sequences of instructions and understand the need to be precise when framing and sequencing instructions.
Level 6: They develop, try out and refine sequences of instructions.
Level 7: They develop, test and refine sequences of instructions as part of an ICT system to solve problems.</td><td>Programming enables boundaries to be broken and new things to be created.
Good programming is efficient – it uses resources (code) already prepared.</td><td>Tutorial1 pp9-18
Making things move
Teacher resources: http://www.tech-faq.com/java.shtml
Keywords include:
class
method
code
http://www.greenfoot.org</td><td>1.1a
1.1b
1.3a
2.2a
2.2e</td></tr>
<tr><td></td><td colspan="5">Starter: Quiz on keywords of last lesson.
Pupils here, focus on the movement of an actor, in this case a ladybird and continue their learning with a copy of their previous work.
Pupils make the bug move forward, backward, left, right, rotate, use a Mover class to that the bug can move in a circle, the bug around when it hits the edge of the world.
Extension: Add a few static actors to the world, and make the bug go around the screen missing the other actors.
Plenary: Oral quiz of the keywords.
Download software to use at home from the downloads page http://www.greenfoot.org</td></tr>
</table>

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
3/7	That programming can determine the graphical user interface through using images. That events comprise of logical steps that can be broken down. Integrate image-editing software into the Greenfoot environment. Use the getbackground() method and importing a class. java.awt.color; use a loop to make decisions; repeating creating an oval to draw the stars.	Level 5: They use ICT to structure, refine and present information in different forms and styles for specific purposes and audiences Level 6: They combine information from a variety of ICT-based and other sources for presentation to different audiences. Level 7: They design ICT-based models and procedures with variables to meet particular needs. Assessed deliverables: Tutorial 3 Stepwise Refinement task.	Programming enables us to control what we can do on screen – something we can all take for granted.	Teacher resources: http://www.tech-faq.com/java.shtml Tutorial1 pp19-28 About Backgrounds Keywords include: image-editing software class loop graphical user interface GUI Look at the Java documentation on the Greenfoot website: http://www.greenfoot.org/doc/javadoc	1.1a 1.1b 2.2a 2.2e
	Starter: discuss the successes in using Greenfoot at home; view successful projects from the previous lesson. Pupils experiment with different ways in which to apply a background to their world. Pupils use Photoshop to draw an image for their background. Pupils are then expected to set this background image in Greenfoot. Actors can then be added to this background image. Pupils are also shown how to program their own background image, in this case, a black sky with stars. Pupils are required to go through each method of 1) creating and 2) programming an image, as they will need to select which to use for their own project. Identify and list some of the methods you have used in your code so far.				

<table>
<tr><th>Lesson Number</th><th>Learning objective
We are learning to
WALT</th><th>Learning outcomes
What I am looking for
WILF</th><th>Rationale
This is because
TIB</th><th>Prerequisite skills of pupil and teacher and resources</th><th>NC, Strategy and QCA references</th></tr>
<tr><td>4/7</td><td>That programming enables events to be created that can be controlled by the user.
Name, use and explain the events:
Mouseclicked
MousedragEnded
MouseDragged
MouseMoved
MousePressed
getmouseinfo()
Understand the concept of “event”.</td><td>Level 5: They create sequences of instructions and understand the need to be precise when framing and sequencing instructions.
Level 6: They develop, try out and refine sequences of instructions and show efficiency in framing these instructions, using sub-routines where appropriate. They assess the validity of these models by comparing their behaviour with information from other sources.
Level 7: They develop, test and refine sequences of instructions as part of an ICT system to solve problems. They design ICT-based models and procedures with variables to meet particular needs.</td><td>The windows operating system is programming based upon events (mouse clicks).
Interactive programs require event driven programming.</td><td>Teacher resources: http://www.tech-faq.com/java.shtml
Tutorial1 pp29-37
Reacting to Mouse Clicks
Keywords include:
(note – raising the level of computing concepts and language)
Event
Mouseclicked
MousedragEnded
MouseDragged
MouseMoved
MousePressed
getmouseinfo()
Look at the Java documentation on the Greenfoot website: http://www.greenfoot.org/doc/javadoc</td><td>1.1a
1.1b
1.3a
2.2a
2.2e</td></tr>
<tr><td></td><td colspan="5">Starter: Teacher demonstrates duplicating and moving an image. Placing images at locations, and changing images upon a mouse click.
Introduce the concept of an ‘event’. What is an event? In what different ways can we react to an event?
Activities for pupils: duplicating an image of a frog, when it is clicked upon; moving an image when it is clicked upon; clicking on a frog will make it jump, and if you click on the background you get a new frog; placing a frog at the location of a click; moving an image around with the mouse; when mouse is pressed change the image, when mouse is released change the image back to its original image.
Experiment with a mouse click event and program one of your own event.
Plenary: Oral quiz of the keywords.</td></tr>
</table>

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
5/7	That programming enables events to be created that can be controlled by the user. And that, the computer output resulting from an event can trigger other events. Understanding that complex programmes are created by combining simple activities.	Programmes of Study - as previous session. Introduction to two 'helper classes': vector class and smoothmover class. Experimentation with the method, GetKey(). Handling exceptions. Preparing the world to start at an initial state. Writing a new method from scratch.	Computer programs are not written as a single complete entity but they are designed to utilise code already developed.	Teacher resources: http://www.tech-faq.com/java.shtml Tutorial2 pp2-20 Debris.png Information about support classes on the Greenfoot website http://www.greenfoot.org/programming/classes.html Keywords include: Helper classes Vector Parameter Exception errors	1.1a 1.1b 1.3a 2.2a 2.2e
	Starter: discuss types of reactions in games, explosions, getting 'gobbled up' etc. Are there any others? Explanation of parameter. At this level of programming, it is advised that pupils work in 2s or 3s to support each other's understanding. Creating the scene - adding a rock and the debris to a scene. Making debris move by using the helper classes - make a piece of debris fall to the floor and make the debris fly in all directions. From making the debris explode, make the rock explode by invoking voidexplode() and then invoking the explosion with a key press. By changing the code the force of the explosion and the shape of the debris can be modified. Exception messages are also covered in this section, so that the pupil is introduced to errors and how to fix them. The world can also be prepared so that it starts at an existing state, for example, with some rock already there. To read up on support classes on the Greenfoot website. Found at http://www.greenfoot.org/programming/classes.html *note it is not required that the student understands each line, but that they become familiar with what these helper classes do.				

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
6/7	That objects can move via pre-existing methods and newly created ones. Use Boolean logic in the code so that actors appear to behave in response to other objects.	Programmes of Study - as previous sessions. Using variables to represent the properties of actors and objects. Create new methods. For example, (fall), checkfall(), jump(). Modify methods and variables. For example, vspeed and acceleration to change the fall effect.	All programs developed need to have a useful purpose and a demand. That commercial programs have a value.	Teacher resources: http://www.tech-faq.com/java.shtml Tutorial2 pp39-50 Debris.png Information about support classes on the Greenfoot website http://www.greenfoot.org/programming/classes.html Keywords include: Variable Boolean Boolean values: TRUE and FALSE	1.1a 1.1b 1.3a 2.2a 2.2e 4b
	Starter: discussion of variables and Boolean logic. Pupils will be given an initial world to work with here. This consists of a cloud background, and an actor called ‘Pengu’, and another actor, which is the ‘ground’. Running the scenario, with the actors in the world enables Pengu to move the left and right, which have been covered in a previous lesson on movement. Create a fall method. Here Pengu will not take any notice of the ground, we will just make him fall. Use of the constant vspeed and acceleration which changes the speed of the fall, Make Pengu check whether he is on the ground by use of a Boolean method. If he is, (Boolean true), then speed is set to 0. If Pengu is not on ground, (Boolean false) and so the Pengu falls. Adding more ground will enable Pengu to move from ground to ground. Create a jump method. Make Pengu react to the space bar to enable him to jump. Also change the height of the jump. Plenary: Oral quiz of the keywords and celebrate the work of some pupils.				

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
7/7 +	Understand the development life cycle. Develop own graphics. Integrate code and. graphics and apply to new methods. Create own game. Submit to CodePoint competition run by Sun Microsystems and University of Kent http://www.greenfoot.org/codepoint	A well researched, designed and executed programming task. Evidence of: Research Design Development Presentation Evaluation	Game writing is a vocational opportunity. Game writing reflects the processes of all product development in the computing world.	Example scenarios http://www.greenfoot.org/scenarios Keywords: Design cycle Research Design Development Presentation Evaluation Criteria Functionality Playability	1.1a 1.1c 1.2a 1.3a 1.5b 2.1a 2.1d 2.2a 2.2b 2.2c 2.2f 4c 2.4b
	Starter: demonstration of a scenario to stimulate ideas, for example Asteroid Research using http://www.greenfoot.org/scenarios to view scenarios. Design ideas based upon pupil ability and knowledge of code evidenced by simple descriptions and drawings. Development of game making use of features and skills learned throughout the unit. Presentation, both formal to the class and informal; playing with other pupils' games. Evaluation including self-assessments and peer assessments. Pupils to judge the game based upon the competition evaluation criteria: Originality / creativity (Is the submission a new idea, or have we seen it before?) Technical difficulty (How technically challenging is the implementation?) Quality (including quality of appearance/graphics; functionality; correctness) Entertainment value (How much does your entry amuse us? For games, this may be called 'playability'. For simulations or other scenarios, this is 'interest') Competition http://www.greenfoot.org/codepoint				

Scenario 3 Programming with Visual Basic

This unit is designed to introduce programming to pupils who have experienced other pre-coding programming activities. It is based upon the use of VB 2008 Express, which can be downloaded free at http://www.microsoft.com/Express/VB/. The scheme of work is supported by resources which are available from http://computingatschool.org.uk/files/CPinKS3

The unit is designed, resourced and used by Emma Wright, Head of ICT & Computer Science, Harvey Grammar School, Folkestone, Kent.

Reference is also made in the scheme of work to a book called “An introduction to programming using VB2005” by David Schneider. A selection of these programs has been used which can be downloaded from http://www.pearsonhighered.com/schneider/details2.html . “*You will need to buy the book to gain access to the ‘instructor resources’. This book is a fantastic resource for teachers and also contains teacher presentations and working answers to all exercises you wish the students to be guided through.*” Emma Wright.

This project covers a minimum of 7 lessons, which are 60 minutes in duration. You may also decide to extend the duration of the project to accommodate an extended piece of work. For example, pupils developing a pizza point-of-sale (POS) program using the knowledge from previous lessons to create a solution.

*Please note: To protect the school network, schools might choose to run the VB 2008 Express software in a Virtual Machine and disable the pupils’ ability to run executables from their computer.

1 Introduction to programming using VB2008 Express.

2 Fundamentals of programming: Events

3 Fundamentals of programming: Procedures

4 Logical Programming Constructs: IF Statements

5 Logical Programming Constructs: CASE

6 The Do..While Loop

7 The For.Next Loop

Extension project Pizza POS.

The activities and curriculum content can be mapped against many aspects of the National Curriculum for ICT (2007).

National Curriculum mapping

1.1a ✓
1.1b
1.1c
1.2a
1.3a ✓
1.3b
1.3c
1.4a ✓
1.4b
1.5a
1.5b

2.1a ✓
2.1b
2.1c ✓
2.1d
2.2a ✓
2.2b ✓
2.2c
2.2d ✓
2.2e ✓
2.2f

2.3a ✓
2.3b
2.3c ✓
2.4a ✓
2.4b ✓
2.4c ✓

3a
3b
3c
3d
3e

4a
4b ✓
4c
4d
4e
4f

The resources can be downloaded from http://computingatschool.org.uk/files/CPinKS3/VBasic.zip

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
1/7	Know and understand the term Object Oriented Programming. Become familiar with the VB programming environment. - Solution Explorer - Control Box & Push Pin - Properties window - Form Window - Main area - Code Window - Menu/Tool Bar To open, run and close a project. Become familiar with design and code view, objects and properties.	Familiarity lesson. Pupils will not be assessed here on anything created.	Students need an understanding of the programming environment.	Program Planning 2.2 p31. Schneider. Resources at http://computingatschool.org.uk/files/CPinKS3/VBasic.zip 1_Introduction Sample Teacher guidance: 1_Introducing_VB_Express.doc 2_The_Design_Enviroment.doc 3_Your_first_VBE_Project.doc	1.1b 1.1c 2.3c
	Introduction to programming: What is a high level language? What is OO Programming? Describe the pictorial representation of the problem solving process. Describe the art of program planning. Pupil Task 1 Pupils to describe each of the processes, of problem solving and program planning above, using MS Word. Pupils are to open up the software, and get an understanding for the environment. This will include: Visual basic controls Toolbox Common controls. Pupils are to open up the sample program experiment and learn how to open, run and close a project. Pupils can create their own project and experiment with adding basic controls such as: Text boxes Labels Buttons Pupils can be shown how to change properties. Extension: Open up the sample code view and discuss with students. Look at the comments and see if any parts of the code can be understood. Plenary What can be produced using visual basic express? Why does Microsoft give it away for free? Test key terms in lesson.				

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
2/7	The visual basic window consists of a form holding a collection controls for which various properties can be set. An event procedure is executed when something happens to a specified object. To write desired event procedures.	The modification of Form Fun Code to develop new events.	We need to show that by modifying code, students can very easily get used to a new programming language and experience success in making new things happen.	Resources at http://computingatschool.org.uk/files/CPinKS3 2_Fundamentals (2) FormFun1 (not a Schneider resource). (3) FormFun2: Example of how FormFun1 can be modified.	1.1a 1.3a 2.2a 2.2b 2.3c

Fundamentals of programming: Events

Pupils are introduced to the 'Event'. They can use their previous knowledge here and see how, using the program 'Form Fun' how events can be actioned.

Pupil Task 2

Using Form Fun, on board only, pupils are to recreate the program to simulate their own events which include:

Adding a new button

Changing background colour

Changing foreground colour

Making the form increase and decrease in size, by more than the original.

Extension: Pupils are to explore other common controls using the toolbox. Try and add a picture to your form! (hint: PictureBox Control)

Plenary: Pupils are to recap the code for an event procedure, and to learn the syntax: ControlName.PropertyName = PropertyValue by looking at further examples.

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
3/7	To learn what is meant by a general and a sub procedure. A variable declared is *class level* and is available to every procedure in the forms code. Variables declared with a Dim statement inside a procedure, are local to the procedure.	That a pupil can use a procedure and get a program working. That they understand the role of variables. That a program can be given an output that uses the result of a process.	That variables are used to make programs efficient. That procedures are used to deal with a smaller part of what might be a larger problem.	Resources at http://computingatschool.org.uk/files/CPinKS3 3_Procedures (4) 4-1-5 Add two numbers together. (Teacher task) (5) 4-1-5 extension (pupil task)	1.1a 1.3a 2.2a 2.2b 2.3c
	General & Sub Procedures Pupils are introduced to a way in which a complex problem can be broken down into small problems. This is by using a sub procedure. Demonstrate and Discuss: Program 4-1-5, and explain to the pupils the structure of the program and how it is broken down, step by step. Students are to understand why we use variables. Task: Pupils are to code this example of adding two numbers together, which outputs the result in a sentence. Extension : Pupils are then to modify this, to multiply three numbers together. (Answer is (5)4-1-5 extension provided).				

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
4/7 5/7	The relational operators are <,>, =, <>, <=,>=. A condition is an expression involving literals, variables, functions, and operators (arithmetic or logical) that can be evaluated as either true or false. The value of a variable or expression of Boolean data type is either true or false. An IF block decides what action to take depending on the truth values of one or more conditions. A Select CASE block selects from one of several actions, depending on the value of an expression, called the selector.	Pupils can create programs using either IF or CASE. Pupils know when to use either IF or CASE in their programming.	IF and CASE are important constructs which in programming are necessary to understand as part of an introductory course. Pupils can also build upon this knowledge in later courses.	Resources at http://computingatschool.org.uk/files/CPinKS3 4_Decisions (9) 5-2-1 Find large number (10) 5-2-2 Profit/Loss (pupil) (11) 5-3-6 Weather beacon (12) 5-3-7 Seasons	1.1a 1.3a 2.2a 2.2b 2.3c
	Logical Programming Constructs: IF Statements & CASE IF (5-2) Allows a program to decide on a course of action based upon whether certain conditions are true or false. Introduce pupil to the pseudo code and flowchart for an IF block. (p201) Lead pupils through the task (9) Pupil to complete task (10). CASE (5-3) Introduce pupil to the pseudocode and flowchart for a CASE block. (p221) Lead pupils through the task (11) Extension: Pupil to attempt Seasons program. (12) Deciding which code structure they need to select (IF or CASE). Homework: To write your own program using IF or CASE. The program can do anything you wish. Print screens of code and output required.				

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
6/7 7/7	A Do Loop repeatedly executes a block of statements either as long as or until a certain condition is true. The condition can be checked at the top end or at the bottom end of the loop. As various items of data are processed by a loop, a *counter* can be used to keep track of the number of items. A flag is a Boolean variable, used to indicate whether a certain event has occurred. A For Next Loop repeats a block of statements a fixed number of times. The control variable assumes an initial value and increments by one after each pass through the loop until it reaches it terminating value.	Completion of programs which illustrate a Do Loop and a For Next Loop. That pupils understand the difference between the two varieties of loops and know in what circumstances to implement either variety.	Do Loops and a For Next Loops are important constructs which in programming are necessary to understand as part of an introductory course. Pupils can also build upon this knowledge in later courses.	Resources at http://computingatschool.org.uk/files/CPinKS3 5_Repetition (13) 6-2-2 Phone number (14) Food Prices (pupil) (15) 6-3-4 Multiplication table. 6_Arrays (16), (17) and (18)	1.1a 1.3a 2.2a 2.2b 2.3c
	A loop, one of the most important structures in Visual Basic, is used to repeat a sequence of statements a number of times. At each repetition, or pass, the statements act upon variables whose values are changing. The Do…While Loop Introduce pupil to the Pseudocode and flowchart for a Do…While Loop. (p248/251) Lead pupils through the task (13) Pupils to attempt on their own (14) The For… Next Loop Introduce pupil to the Pseudocode and flowchart for a For … Next Loop. (p278) Lead pupils through the task (15) Extension: Change the multiplication table to make calculations of your choice. Plenary: Introduce the concept of nested For-Next Loops (multiplication example is one of these), and discuss implications for structure of programs – see p283. Homework: When is it best to use a Do …While Loop as opposed to a For Next Loop? Write answer in Word giving examples of when to use each.				

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
8+	To work through the development an application unaided. To use existing skills and learn how to apply them.	Pupils to develop their own program (fully or part working) using the concepts learned in previous lessons.	Pupils need to experience software development first hand, and be creative.		2.1a 2.1c 2.2b 2.2d 2e 2.3a 2.4a 2.4b 2.4c 4b
	Project Brief Pupils are expected to use the programming skills they have developed to create their own solution for a Pizza Point of Sale Software. This may include an adding function as well as a simple receipt for the more able (see examples 6 & 7 above. Design Pupils need to think about their design before programming. How could skills learned previously help them in creating this application? Pupils are to draw on paper what their application is going to look like. The teacher might like to check this prior to the dev elopement stage. Level 7 pupils are required to document flow charts for specific processes within their system. L7 Also requires an evaluation and re-development based on assessment. Level 8: Create a user guide and systems manual ensuring coverage of areas in which the user might find some operations difficult. Development Pupils are expected to then develop their solution, subject to their teachers' approval. Output Pupils are to print screen and annotate their work, also printing out their code (with comments). Extension: To look at teacher example, and reverse engineer some new skills into their own work.				

Scenario 4 A game business using Game Maker

This unit is designed to introduce programming to pupils through the open source game scripting package Game Maker (http://www.yoyogames.com/gamemaker).

“Using easy to learn drag-and-drop actions, you can create professional looking games within very little time. You can make games with backgrounds, animated graphics, music and sound effects, and even 3D games! And when you've become more experienced, there is a built-in programming language, which gives you the full flexibility of creating games with Game Maker. What is best, is the fact that Game Maker can be used free of charge.” (Game Maker website, 2009)

Game Maker has a standard windows interface which allows you to make exciting computer games, using easy to learn drag-and-drop actions. A beginner can create professional looking games with no programming knowledge within very little time. You can make games with backgrounds, animated graphics, music and sound effects. Within the advanced facility there is a built-in programming language, which gives the more experienced student flexibility of creating games with Game Maker. The standard version allows you to do anything you want with the games you produce, you can even sell them! Also, if you register your copy of Game Maker, you can unlock extra functions, which extend the capabilities of the program. Game Maker comes preloaded with a collection of freeware images and sounds to get you started.

This scheme of work was created by Liz Crane for Oaklands Catholic School, Waterlooville, Hampshire. It combines the game authoring activities and business related skills and knowledge activities to form a 15-lesson structure that enables pupils to experience many aspects of ICT and focus upon those elements that interest them the most. The intention is to offer level 5 opportunities for all with extension opportunities for some.

Over the 7 lessons of the modules pupils experience the cycle of activities that include the design, production, business planning, marketing and sales management. It requires pupils to have experienced each of those activities in a teaching module. This module focuses upon the combination of applications and is most suitable for year 9 as it prepares pupils for project work associated with accredited courses in key stage 4.

“Forget about Flowol and creating a simulation to operate a Flume Ride, use Game Maker for students to create their own game. Creating a game satisfies the criteria for students to be able to sequence events and put commands in right order. The assignment is written for students to set up their own business and it incorporates cash flow, database, web pages and online form. The students practically enjoyed the challenge and even worked on it at home. The less able pupils were able to access it and the girls came top when completing the game as they took it at a slower pace and worked out the commands methodically. You can download the simple version of Game Maker free of charge.” Liz Crane, Oaklands Catholic School.

WALT

We are learning to…
should be the skills, knowledge and understanding of the lesson
(perhaps also “attitudes”);

WILF

What I’m looking for…
the assessment for learning or assessment for teaching statements
(supports Assessment for Learning);

TIB

This is because… the rationale for teaching
how to copy using absolute cell referencing, the essence of “computing”.

United by the Cross

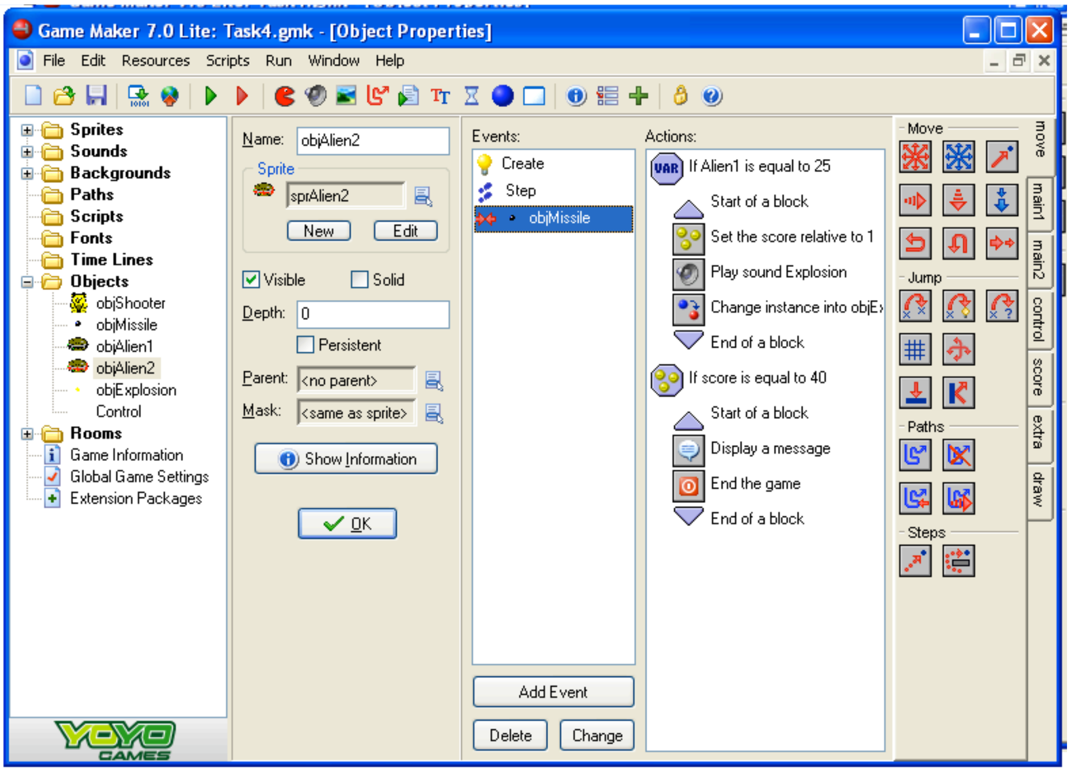
Oaklands Catholic School,
Waterlooville, Hampshire

Lesson Number	Learning objective We are learning to WALT	Learning outcomes What I am looking for WILF	Rationale This is because TIB	Prerequisite skills of pupil and teacher and resources	NC, Strategy and QCA references
1/7	Creating a game – introducing Game Maker.		Game making introduces ideas relating to story-boarding, design specification	Game Maker http://www.yoyogames.com/gamemaker	1.1a 1.1b 2.2a 2.3c
2/7	Create a game that has interest (audience) and sophistication (of algorithm)	a sequence, loop and different things happen depending on changes in variables; more efficiency through adding a loop, sub routine	Effective programming meets the needs of the audience; efficient programming uses structures		
3/7	Creating a cash flow – introducing the business model		Entrepreneur-ship		
4/7	Create a spreadsheet model. Consider the layout and content of the model. Use a set of rules to predict values and solve problems	solving problems by posing and answering "what if" questions; predicting outcomes	Computer models form the basis of many businesses		
5/7	Creating a Database	prepare a flat file; using a data capture as a planning tool; test the data set; plan and then create a data entry form			
6/7	Creating a Web page and an online form	website to meet the needs of the audience; a specific purpose			
7/7	Create a set of criteria to judge their own work and that of their peers.	agree the criteria; evaluate others' work; reflect on evaluations		Prompt sheet GoodGame.doc	

Game Maker in Key Stage 3

The following review describes the use of Game Maker to support pupils' independent work in a computing environment in Key Stage 3, written by Claire Johnson, ICT Subject Leader, Westgate School, Winchester.

Game Maker is a game development tool which is increasingly used in UK schools, because of its accessibility to beginners, while allowing more advanced users scope to create more complex games. This makes it a useful addition to the Key Stage 3 ICT curriculum, where it can be used to cover the sequencing instructions strand of the National Curriculum Programme of Study.



Game Maker 7.0 Lite: Task4.gmk - [Object Properties]

Name: objAlien2

Sprite: sprAlien2

☑ Visible ☐ Solid

Depth: 0

☐ Persistent

Parent: <no parent>

Mask: <same as sprite>

Events:
- Create
- Step
- objMissile

Actions:
- If Alien1 is equal to 25
- Start of a block
- Set the score relative to 1
- Play sound Explosion
- Change instance into objE
- End of a block
- If score is equal to 40
- Start of a block
- Display a message
- End the game
- End of a block

"In Game Maker, users add events and drag and drop actions to control objects to create 2D games. Game Maker allows the creation of many types of games, including maze and platform games, first and third person shooters and even simple 3D games. It takes about 10 hours to learn the program and then to create a reasonably satisfying game (including the creation of graphics, audio, splash screen, and player instructions).

Users select from a set of standard action libraries for movement, control structures, drawing, scoring and so on. Advanced users can extend the drag and drop functionality of Game Maker by creating new actions, or by using the built in Game Maker Language (GML), which allows users to write and add scripts as actions. Even if users do not learn GML, they still learn about basic syntax and OOP.

There is a popular on line community where games can be uploaded and shared – and this real audience is motivating to users, as well as giving them access to forums, resources and tutorials. A lite version is freely downloadable from *http://www.yoyogames.com*. A registered version is £20, or sold as educational site licences in 5 user packs (£50). Alongside Scratch, Game Maker seems to be the game authoring program of choice featured in recently published school textbooks (ICT4Life Book 1 (Year 7); ICT Interact (Year 8) . A scheme of work supported by video tutorials, lesson plans and pupil resources is available from www.teach-ict.com. Payne-Gallway's 'Basic Projects: Game Maker' [ISBN: 9781905292578] includes accessible and reasonably challenging tutorials for a breakout game and a space invaders game. Here pupils are shown how to write a simple script and attach it to an action. As always, teachers need to skill themselves up to a reasonable level before they can really start to use this tool to develop a more challenging 'programming' experience for KS3/4 pupils.' The Game Makers Apprentice' by Mark Overmars and Jake Habgood is recommended reading."

Claire Johnson, June 2009

Scenario 5 Games and game authoring

Games can be classified in a number of different ways, work conducted by Futurelabs (2005) categorised games in education based upon their characteristics. The works by Appleby, T (2005) classified games based on their genre, like any typical taxonomy, a game genre must have certain constants, that is, things that remain the same. Globally, all games have obstacles to overcome. Therefore one can define their genres within the way that these obstacles are completed. While not a comprehensive list computer games can normally be described by one of the following classifications: action; adventure; simulation; role play; and strategy.

Action games are perhaps the most basic of gaming genres, and certainly one of the broadest. In an action game, players use quick reflexes and timing to overcome visual obstacles. Common examples include Pin Ball where the action of a player moves the ball and Pac Man where the success of the game is dependent upon the quick reactions of the Game Player.

Adventure games were some of the earliest games created, originally text based stories. Over time, graphics have been introduced to the genre and the interface has evolved. Adventure games usually normally require the player to solve various puzzles by interacting with people or the environment, most often in a non-confrontational way. It is considered a "purist" genre. Because they put little pressure on the player in the form of action-based challenges or time constraints, adventure games have had the unique ability to appeal to people who do not normally play video games. Examples include Zore and Ace Attorney.

Simulation games can control a number of different situations the most Common are Building, Management and Transport. In a Simulation Game the player needs to build, develop, create or manage projects or communities with little of few resources. Sim games have evolved considerably over recent years and with easy access to the internet many Sim games allow you to compete against real competitors or groups through game networks. Examples include Sim City, Populous and Theme Hospital.

Role Play games cast the player in one or more adventure roles. The adventure will then have a skills set which the player will use a range of different situations and scenarios. As games have evolved from simple RPG to highly graphical interfaces many of the games include locations such as towns, buildings and castles. Role play games include Dragon Quest, Kingdom Hearts and with the emergence of the internet online role-playing games such as World of Warcraft.

Strategy games require careful and skilful planning from the game player. Many strategy games originate from the concept of board games. As games have evolved there are now many different types of strategy games, including artillery, war tactical and defence. Similar to other games the evolution of the internet has led to a range of real time games where players can play with others over the internet. Examples of strategy games include Scorched Earth and Close Combat.

The classification of a game can be subjective while It is also possible to have a hybrid where the games cross a number of different classifications an example of this is a SIM game where there is an aspect of collaboration, challenge and identity within the one game. Other examples include web based games such where there is an additional collaboration and communication aspect to the game.

Keywords

Keywords that will be encountered within this teaching unit include:

Plan
Script
Authoring
Classification
Control
Programming
Genre
Scenario
Implement
Sequence
Interface
Audience
Process
Automate
Platform

The scenario is based upon: students have been asked by Nintendo to create a computer game for young children that has an educational value. Students will identify the initial problem, analyse the problem, design a solution, implement a computer game and evaluate the game against the original objectives.

Lesson Number	Learning objective	Learning outcomes	Rationale This is because TIB	Prerequisite skills of pupil and teacher, resources and	NC Strategy references (2008)
1	Introduction & Problem Identify the problem and explore possible solutions.	Identify the initial problem – what are the objectives of the game. That you can arrange a sequence of instructions for a set of traffic lights into the correct order	Importance of game authoring to make computers do what we wish them to do.	Lesson 1 (http://dmarshall.moodle4free.com/course/view.php?id=14) Keywords display Teacher's dictionary.	1.1a, b, c 1.3a 1.4a, b 2.1a 2.2a 2.3a, b, c 3a 4a,b, e
2	Analysis Lesson 1 Analyse the inputs, processes and outputs of the computer game.	That a computer game is a system and has Inputs Processes and Outputs. Recognise the advantages of ICT systems over manual systems.	To understand how computer games are designed written and work.	Lesson 2 (http://dmarshall.moodle4free.com/course/view.php?id=14) Keywords display Teacher's dictionary.	1.1a, b, c 1.3a, b 2.1a, b,d 2.2b, e 2.3a, b,c
3	Analysis Lesson 2 Analyse the inputs, processes and outputs of the computer game.	Indentify Inputs Processes and Outputs. Recognise the advantages of ICT systems over manual systems.	To understand how computer games are designed written and work.	Lesson 3 (http://dmarshall.moodle4free.com/course/view.php?id=14) Keywords display Teacher's dictionary.	1.1a, b, c 1.3a, b 2.1a, b,d 2.2b, e 2.3a, b,c

Lesson Number	Learning objective	Learning outcomes	Rationale This is because TIB	Prerequisite skills of pupil and teacher, resources and	NC Strategy references (2008)
4	Initial Designs Create/design the stages of your computer game.	Paper based designs of game environment. Storyboard of main game events.	To design a game that meets the user's needs. Structured approach to games development.	Lesson 4 (http://dmarshall.moodle4free.com/course/view.php?id=14) Keywords display Teacher's dictionary.	1.1a, b, c 1.3a, b 1.5a, b 2.1a 2.2a,c, d, e 2.3a, b,c
5	Design – Test Plan Create a test plan to test the computer game.	Plan at least 4 tests, reason, example data and predicted outcome.	To understand that all computer programmes and games need to be fully tested.	Lesson 5 (http://dmarshall.moodle4free.com/course/view.php?id=14) Keywords display Teacher's dictionary.	1.3a, b,c 1.5a, b 2.1d 2.2c,e 2.3a
6	Implementation Start to implement your computer game	Create a number of game objects. Attach events to the objects	To understand that objects require events. Structured approach to games development.	Lesson 6 (http://dmarshall.moodle4free.com/course/view.php?id=14) Keywords display Teacher's dictionary.	1.1a, b,c 1.3b, c 2.2a, b,c,d, e,f 2.3a 3a

Sample teaching unit – computer programming

Lesson Number	Learning objective	Learning outcomes	Rationale This is because TIB	Prerequisite skills of pupil and teacher, resources and	NC Strategy references (2008)
7/8/9	Implementation Continue to Implement your computer game (Lessons 7/8 and 9 are optional)	Create a number of game objects. Attach events to the objects	To understand that objects require events. Structured approach to games development.	Lesson 7/8/9 (http://dmarshall.moodle4free.com/course/view.php?id=14) Keywords display Teacher's dictionary.	1.1a, b,c 1.3b, c 2.2a, b,c,d, e,f 2.3a 3a
10	Evaluation Evaluate your game against the original objectives	To evaluate the game against the initial problem/objectives.	Computer programmes and games have objectives which they have been designed to meet.	Lesson 10 (http://dmarshall.moodle4free.com/course/view.php?id=14) Keywords display Teacher's dictionary.	2.4a, b,c

Homework Tasks

Task 1 Evaluate your favourite Computer Game

Students are asked to evaluate their favourite computer game these can be on any platform, students are asked what target audience the game is designed for, what is good and what can be improved students are given a template to help them organise their thoughts and ideas.

Task 2 Invent your own computer game

Students are asked to think about designing a new computer game. They need to think carefully about whom their game is for and their chosen genre. They are then to think about the characters in their game and are given a template to help them organise their thoughts and ideas.

Task 3 Scenario Mind Map

Students should create a Mind Map about their chosen scenario or theme. The concept of Mind Maps should be demonstrated with the main 'areas' being the roots of the tree. Those who have access to a computer may wish to use a Mind Map package such as Inspiration or Bubble.

Resources for programming (alphabetical)

Alice
An Educational Software that teaches students computer programming in a 3D environment
A revolution in computer science pedagogy

Alice: This is a really deep learning environment from Carnegie Mellon. Can be used from an early age up to University level. http://www.alice.org/

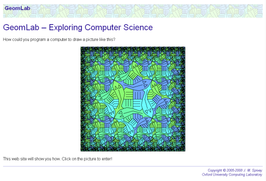
Computer Science Unplugged: Wonderful set of exercises from Canterbury University in New Zealand. http://csunplugged.com/

CS Inside: Lots of downloadable exercises demonstrating a variety of computing concepts from Glasgow University. http://csi.dcs.gla.ac.uk/index.php

CS4Fn: Free magazine and excellent supporting website from Queen Mary College, London. http://www.cs4fn.org/ Recently launched another free magazine Audio, looking at computing technology in music. CS4Fn editor, Paul Curzon has written a series of excellent articles demonstrating basic computing concepts, downloadable from http://www.dcs.qmul.ac.uk/%7Epc/research/education/puzzles/reading/

Details of Manchester University animation competition for schools using Alice, Scratch or Flash are at http://www.cs.manchester.ac.uk/Animation09/

FreePascal: http://www.freepascal.org/



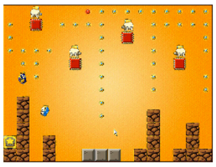
Functional Programming: If you wish to introduce a different programming paradigm http://web2.comlab.ox.ac.uk/geomlab/index.html is a wonderful introduction to functional programming, developed as part of the gifted and talented initiative. It can make an excellent project for an immersion / extension activity. The environment and worksheets can be downloaded from the site.

GameMaker: Kids love it! http://www.yoyogames.com/gamemaker

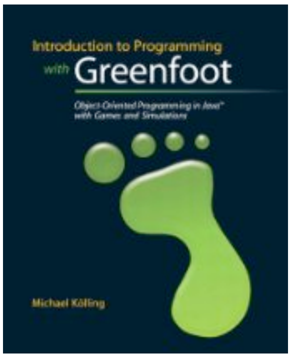
Details of the supporting book: http://book.gamemaker.nl/

An excellent introductory site with school level tutorials is http://www.mindtools.tased.edu.au/gamemaker/default.htm

Gamestar Mechanic



GameStar Mechanic: Another site worth keeping an eye on, particularly if you are interested in the notion of gaming as a vehicle to teach computing is Robert Torres GameStar Mechanic. Currently in private beta but due for public release soon. "In Gamestar Mechanic, players learn the fundamentals of game design. Within the game, players take the role of "game mechanics" in a steampunk world where the rules and elements of games have come to life as creatures. Players use these creatures to repair broken games and create new ones. As an apprentice game mechanic, players prove their expertise by completing a series of increasingly complex game design challenges…." http://www.instituteofplay.org/node/162/


Introduction to Programming
with Greenfoot
Object-Oriented Programming in Java™ with Games and Simulations
Michael Kölling

Greenfoot: Environment for introducing object oriented programming from the University of Kent. http://www.greenfoot.org/index.html

The Introduction to Programming with Greenfoot by Michael Kölling is an entry-level introduction that systematically takes the learner through the basics of Object Oriented Programming in a clearly illustrated and pedagogically sound way and is suited to key stage 3 teaching.

http://astore.amazon.co.uk/pgce-21/detail/0136037534

Kara, the programmable ladybug

© 2003 - 2006

www.swisseduc.ch/informatik/karatojava

Kara

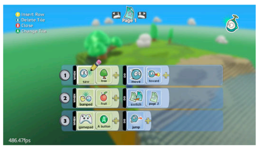
idea, concept & programming
Jürg Nievergelt
Werner Hartmann
Raimond Reichert
Markus Brändle
Tobias Schlatter

contributions
Philipp Boksberger
Remo Meier
Reto Lamprecht
Nicole Völki
Samuel Zürcher

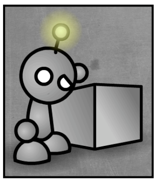
actor state machine editor ...

Kara: An alternative approach to teaching programming via a language or a ‘microworld’ is Kara. What makes Kara special is that its programs are finite state machines created in a graphical program editor. The program does not include commands in the sense of a high level language but rather a series of instructions based on the inputs from the grid and the next state in the program. Developed by SwissEduc they explain the approach thus: “Two characteristics make Kara attractive for introductory courses: Finite state machines are easy to understand, which means the time needed to get started is minimal. And with Kara, you work in a simple, easy-to-use environment without having to deal with the complexities of modern programming environments.” http://www.swisseduc.ch/compscience/karatojava/ The approach is elaborated in this thesis: http://www.asiplease.net/computing/kara/index.htm

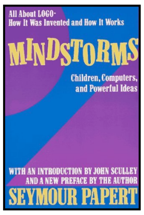
Programming as a Creative Medium



Kodu: Microsoft’s 3D Game Creator for the Xbox, released earlier this year can be found at http://research.microsoft.com/en-us/projects/kodu/ A version for the PC is being developed which could provide an excellent resource for early secondary pupils. Input is via an Xbox controller. Blog http://community.research.microsoft.com/blogs/kodu/default.aspx



LightBot can be found on lots of game sites. A marvellous game for challenging pupils to develop logical and sequencing skills involving just a couple of functions. Levels get increasingly complex. http://armorgames.com/files/games/light-bot-2205.swf A version is now available for download onto iPhones.


All About LOGO-
How It Was Invented and How It Works
MINDSTORMS
Children, Computers, and Powerful Ideas
WITH AN INTRODUCTION BY JOHN SCULLEY
AND A NEW PREFACE BY THE AUTHOR
SEYMOUR PAPERT

Logo: The language that started it all! Various free versions available. Here is one - http://www.numeracysoftware.com/freeMSWlogo.html

Processing is an open source programming language and environment for people who want to program images, animation, and interactions. It is used by students, artists, designers, researchers, and hobbyists for learning, prototyping, and production. It is created to teach fundamentals of computer programming within a visual context and to serve as a software sketchbook and professional production tool.

The home of the Programimate

Page Contents:

- What is Programimate
- About this Website

Programimate supports a range of languages that currently includes:

- Java,
- VB.NET,
- VB6.

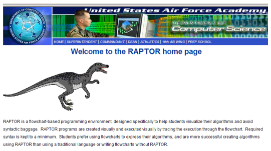
The programming problems on this site are available in all of the above languages.

Please note log in and registration

An excellent resource for introducing basic computing structures. From Andrew Scott and colleagues at Glamorgan University. A web based resource that allows you to construct flowcharts and watch as the computer executes your commands. Code is automatically generated in 3 languages allowing comparison with the flowchart whilst avoiding the problem of initial syntax errors. http://www.comp.glam.ac.uk/pages/staff/asscott/progranimate/

Python: http://www.python.org/ Good entry level language with minimal syntax problems. Various projects to introduce this to pupils. Look at the course http://www.livewires.org.uk/python/home, the free book http://www.briggs.net.nz/log/writing/snake-wrangling-for-kids/

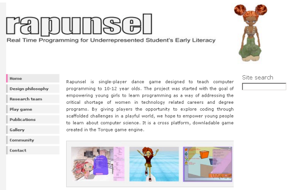
and the robot environment RUR-PLE http://rur-ple.sourceforge.net/


United States Air Force Academy
DEPARTMENT OF Computer Science
HOME | SUPERINTENDENT | COMMANDANT | DEAN | ATHLETICS | 10th AB WING | PREP SCHOOL

Welcome to the RAPTOR home page

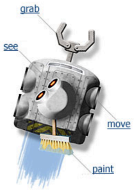
RAPTOR is a flowchart-based programming environment, designed specifically to help students visualize their algorithms and avoid syntactic baggage. RAPTOR programs are created visually and executed visually by tracing the execution through the flowchart. Required syntax is kept to a minimum. Students prefer using flowcharts to express their algorithms, and are more successful creating algorithms using RAPTOR than using a traditional language or writing flowcharts without RAPTOR.

Raptor: Developed by the US Air Force Academy to help visualise algorithms and minimize syntax baggage. http://www.usafa.af.mil/df/dfcs/bios/mcc_html/raptor.cfm



Rapunsel is single-player dance game designed to teach computer programming to 10-12 year olds. The project was started with the goal of empowering young girls to learn programming as a way of addressing the critical shortage of women in technology related careers and degree programs. By giving players the opportunity to explore coding through scaffolded challenges in a playful world, we hope to empower young people to learn about computer science. It is a cross platform, downladable game created in the Torque game engine.

Rapunsel: A project due to come to fruition shortly, developed by a team lead by Ken Perlin of New York University is Rapunsel (http://rapunsel.org/) A single-player dance game designed to teach computer programming to 10-12 year olds. It will be a cross platform, downloadable game. More info at http://www.maryflanagan.com/rapunsel/about.htm



RoboMind: Wonderful programmable robot environment from University of Amsterdam. Easy entry level for young children. http://www.robomind.net/en/index.html

Scratch: Great resource from MIT for introducing programming http://scratch.mit.edu/ Fast becoming the standard introductory programming environment in schools. Developed by Mitch Resnick's team Scratch builds on the approach pioneered by Lego Mindstorms, using colour coded snap together blocks to create sequences of instructions. It thus makes programming accessible to children who have limited literacy skills in that it removes the syntax barrier. See http://info.scratch.mit.edu/Educators for further resources to support teaching.

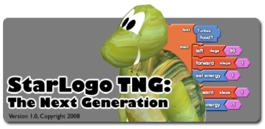
Small Basic V0.3 is Now Available

Small Basic: http://msdn.microsoft.com/en-us/devlabs/cc950524.aspx For those who want to teach programming using 3rd generation BASIC – it's simplicity is compelling

Etoys is . . .

- an educational tool for teaching children powerful ideas in compelling ways
- a media-rich authoring environment and visual programming system
- a free software program that works on almost all personal computers

Squeak: Alan Kay's original educational software http://www.squeakland.org/. Installed on the OLPC and the software behind developments such as Scratch. See Alan Kay demonstrate it in this TED Talk: http://www.ted.com/talks/lang/eng/alan_kay_shares_a_powerful_idea_about_ideas.html



StarLogo TNG: Fantastic for creating models, with lots of samples to use in other subjects too. Simply the best free resource for introducing modelling in the ICT curriculum. Some great potential for cross curricular work. Another development from Mitch Resnick and MIT's Media Lab, this new version introduces snap together blocks and 3D For a ready to use resource, including presentation and sample code based on a StarLogo Tutorial for modelling the spread of an epidemic see: http://www.computingatschool.org.uk/files/conference2009/epidemic.pdf and http://www.computingatschool.org.uk/files/conference2009/davies.zip

VB.Net: Code Rules, a course to introduce pupils to programming in VB.Net is available from http://msdn.microsoft.com/en-us/beginner/bb308755.aspx There is a link there to download VB.Net Express

References

Alice http://www.alice.org [April 2009]

Blackwell, AF and Green, TRG (1999) *Investment of Attention as an Analytic Approach to Cognitive Dimensions* in T. Green, R. Abdullah & P. Brna (Eds.) Collected Papers of the 11th Annual Workshop of the Psychology of Programming Interest Group (PPIG-11), pp. 24-35 http://www.cl.cam.ac.uk/~afb21/publications/PPIG99.html [April 2009]

Computer Science for Fun http://www.cs4fn.org [April 2009]

Church, L and Whitten, A (2009) *Generative Usability; Security and User Centered Design beyond the Appliance* Oxford, UK: New Security Paradigms Workshop (to be published)

DCSF (2008) *Assessing pupils' progress in ICT at Key Stage 3* London, UK: Department for Children, Schools and Families

DfES (2002a) *Sample Teaching Unit for ICT: Year 7, Unit 3* London, UK: Department for Education and Skills http://nationalstrategies.standards.dcsf.gov.uk/node/156743 [April 2009]

DfES (2002b) *Key Stage 3 National Strategy Framework for teaching ICT capability: Years 7, 8 and 9* London, UK: Department for Education and Skills

Game Maker http://www.yoyogames.com [April 2009]

Greenfoot http://www.greenfoot.org [April 2009]

Kölling, M (2009) Introduction to Programming with Greenfoot New York, US: Pearson http://www.greenfoot.org/book [April 2009]

Malone, TW (1980) What Makes Things Fun to Learn? Heuristics for Designing Instructional Computer Games. Paper presented at the *Association for Computing Machinery Symposium on Small and Personal Computer Systems*, Pal Alto, California.

Overmars, M and Habgood, J (2006) *The Game Maker's Apprentice: Game Development for Beginners* Technology in Action http://book.gamemaker.nl [April 2009]

QCA (2007) The National Curriculum 2007 ICT Programme of study for key stage 3 London, UK: Qualifications and Curriculum Authority http://curriculum.qca.org.uk/key-stages-3-and-4/subjects/ict/keystage3 [April 2009]

Wing, J (2006) *Computational Thinking* Communications of the ACM 49 3 http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf [April 2009]

Appendix 1

Level descriptors for the National Curriculum ICT (2008 onward)

Level 4

Pupils combine and refine different forms of information from various sources… …they exchange information and ideas with others in a variety of ways, including using digital communication. They understand the risks associated with communicating digitally, including the security of personal information. They plan and test sequences of instructions. They use ICT-based models and simulations to explore patterns and relationships, and make predictions about the consequences of their decisions. They use ICT to organise, store and retrieve information. They compare their use of ICT with other methods and with its use outside school.

Level 5

Pupils combine ICT tools within the overall structure of an ICT solution. They select the information… …they exchange information and ideas with others in a variety of ways, including using digital communications. They create sequences of instructions and understand the need to be precise when framing and sequencing instructions. They explore the effects of changing the variables in an ICT-based model. They use ICT to organise, store and retrieve information using logical and appropriate structures. They use ICT safely and responsibly. They discuss their knowledge and experience of using ICT and their observations of its use outside school. They assess the use of ICT in their work and are able to reflect critically in order to make improvements in subsequent work. They use appropriate evaluation criteria to critically evaluate the fitness for purpose of their work as it progresses.

Level 6

Pupils plan and design ICT-based solutions to meet a specific purpose and audience, demonstrating increased integration and efficiency in their use of ICT tools. They develop and refine their work to enhance its quality, using a greater range and complexity of information. Where necessary, they use complex lines of enquiry to test hypotheses. They present their ideas in a variety of ways and show a clear sense of audience. They develop, try out and refine sequences of instructions and show efficiency in framing these instructions, using sub-routines where appropriate. They use ICT-based models to make predictions and vary the rules within the models. They assess the validity of these models by comparing their behaviour with information from other sources. They plan and review their work, creating a logically structured portfolio of digital evidence of their learning. They discuss the impact of ICT on society.

Level 7

Pupils design and implement systems. They are able to scope the information flow required to develop an information system. They combine information from a variety of ICT-based and other sources for presentation to different audiences. They identify the advantages and limitations of different information-handling applications. They select and use information to develop systems suited to work in a variety of contexts, translating enquiries expressed in ordinary language into the form required by the system. They develop, test and refine sequences of instructions as part of an ICT system to solve problems. They design ICT-based models and procedures with variables to meet particular needs. They consider the benefits and limitations of ICT tools and information sources and of the results they produce, and they use these results to inform future judgements about the quality of their work. They make use of audience and user feedback to refine and enhance their ICT solutions. They take part in informed discussions about the use of ICT and its impact on society.

Level 8

Pupils independently select appropriate information sources and ICT tools for specific tasks, taking into account ease of use and suitability. They design successful ways to collect and prepare information for processing. They design and implement systems for others to use. They take part in informed discussions about the social, economic, ethical and moral issues raised by ICT.

Exceptional performance

Pupils evaluate software packages and ICT-based models, analysing the situations for which they were developed and assessing their efficiency, ease of use and appropriateness. They suggest refinements to existing systems and design, implement and document systems for others to use, predicting some of the consequences that could arise from the use of such systems. When discussing their own and others' use of ICT, they use their knowledge and experience of information systems to inform their views on the social, economic, political, legal, ethical and moral issues raised by ICT.