iCompute teaching

helping schools to inspire and shape creative computing

iCompute

For iPad

Year 6 - Unit 2

bett awards 2014 FINALIST bettawards.com

bett AWARDS 2015 FINALIST

iPad

Computing with iPads

Year 6 – Unit 3

Overview

This unit duplicates iProgram (Year 6) from the whole-school pack, providing the option to teach algorithms and programming using iPads instead of pcs. Using the context of art and drawing, the children will be engaged in creatively developing simple animations using the Pyonkee app – a visual programming language based on Scratch 1.4.

Objectives

- See p2 for a detailed breakdown of lesson assessment focuses and associated success criteria.

2014 Computing Programme of Study

- design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts
- use sequence, selection, and repetition in programs; work with variables and various forms of input and output
- use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs
- understand computer networks including the internet; how they can provide multiple services, such as the world wide web; and the opportunities they offer for communication and collaboration
- use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content
- select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information
- use technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour; identify a range of ways to report concerns about content and contact.

Assessment

p18 contains a record of progress pro-forma

Apps

Pyonkee (free) app available here:

- Pyonkee - Visual Programming with iPad

Other Resources

Scratch reference guide download available here:

- http://info.scratch.mit.edu/support/reference_guide_1.4

Royalty free sound files available here:

- http://fxhome.com/sound-effects
- http://www.grsites.com/archive/sounds/

Curriculum Links

- Mathematics
- Art/Design

Objectives

Lesson	Title	NC Links	Objectives	Vocabulary	Curricular Links	Success Criteria
6.3.1	iControl	• Work with variables and various forms of input and output • select, use and combine a variety of software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information	• To understand the difference between games and simulations • To identify the various inputs that computer games can use	control; input; output; simulation	Science	• The children can talk about which games are set in imaginary worlds and involve challenge and those that mimic real-life situations • The children identify a range of input devices used in playing computer games
6.3.2	iGame	• design, write and debug programs that accomplish specific goals; solve problems by decomposing them into smaller parts • Use sequence, selection and repetition in programs; work with variables and various forms of input and output	• To program a computer game by sequencing conditional statements	control; input; output; process; condition; statement; if; then	Art/Design	• The children program a simple 'pong' style computer game using if..then programming structures
6.3.3	iPlan	• design, write and debug programs that accomplish specific goals; solve problems by decomposing them into smaller parts	• To understand that the behaviour of a computer program should be planned • To understand that programs are developed according to a plan	design; plan; logical operators (greater than, less than, equal to, less than and equal to, greater than and equal to); variables	Mathematics	• The children record the sprites, backgrounds, interactions and scoring rules for a simple 'Angry Birds' style computer game
6.3.4	iCode	• design, write and debug programs that accomplish specific goals; solve problems by decomposing them into smaller parts • Use sequence, selection and repetition in programs; work with variables and various forms of input and output • use search technologies effectively, appreciate how results are selected and ranked, and be discerning in evaluating digital content • select, use and combine a variety of	• To program an algorithm according to a plan	algorithm; plan; sprite; costume; variable; iteration (repeat; forever; while)	Art/Design	• The children create the sprites necessary for their game • The children program their sprites to move

Lesson	Title	NC Links	Objectives	Vocabulary	Curricular Links	Success Criteria
		software (including internet services) on a range of digital devices to design and create a range of programs, systems and content that accomplish given goals, including collecting, analysing, evaluating and presenting data and information				
6.3.5	iDevelop	• design, write and debug programs that accomplish specific goals; solve problems by decomposing them into smaller parts • Use sequence, selection and repetition in programs; work with variables and various forms of input and output; generate appropriate inputs and predicted outputs to test programs	• To develop a program according to a plan	algorithm; plan; sprite; costume; variable; iteration (repeat; forever; while)	Art/Design	• The children add variables and program the scoring rules necessary to progress in their game • The children add the backgrounds and background changes necessary for a change in level in their game
6.3.6	iDebug	• design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts • use sequence, selection, and repetition in programs; work with variables and various forms of input and output • use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs	• To develop strategies for testing and debugging computer programs	test; debug; amend; systematically		• The children can identify mistakes in their own and other's computer programs • They make changes to their code when things do not execute as expected

Preparation

- Read the lesson plans
- Check the Scratch reference guide glossary of terms to make sure you are using the correct vocabulary
- Spend an hour or so familarising yourself with the software you will be using and the application's interface

Resources

- iPads
- Ensure that the app is installed on each iPad you will be using
- Worksheets for each lesson – entitled: Worksheet<year.unit.lesson> (eg.Worksheet6.3.1)
- Support materials for each lesson – entitled Resource <year.unit.lesson> (eg. Resource6.3.1)

Links

Before you start, you may find these weblinks useful.

- Pyonkee can be downloaded at Pyonkee - Visual Programming for iPad
- Scratch Reference Guide: http://info.scratch.mit.edu/Support/Reference_Guide_1.4

Updates

If any links are not working, visit the iCompute website which contains up-to-date links:

http://www.icompute-uk.com/Links.html

Resources: Resource6.3.1; Resource6.3.1 Image Folder; Worksheet6.3.1; Teacher6.3.1(answers); scissors; glue; iPads

Objectives

- To understand the difference between games and simulations
- To identify the various inputs that computer games can use

Success Criteria

- The children can talk about games they know that are set in imaginary worlds and involve challenge and those that mimic real-life situations
- The children identify a range of input devices used in playing computer games

Vocabulary

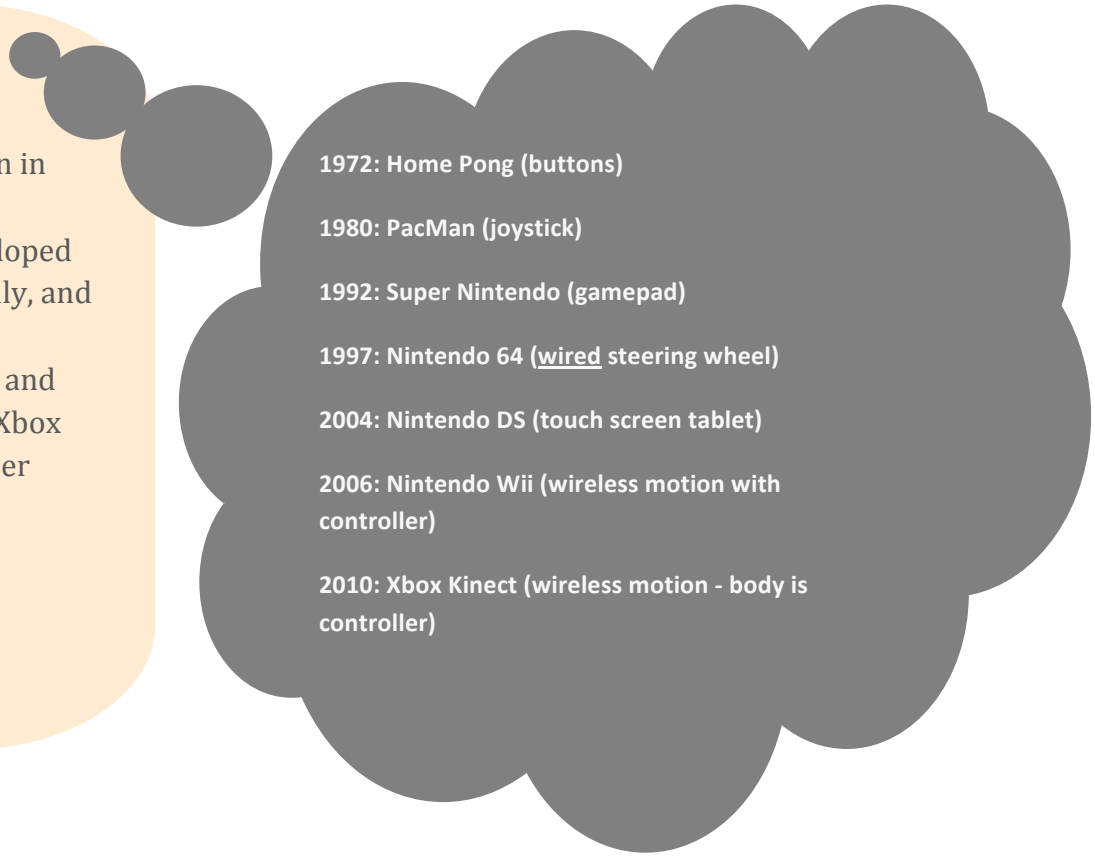
control; input; output; simulation

Step 1

- Talk about the computer games the children know of and have used
- Ask them to identify their favourites and explain where they are set and how you progress in the games
- Discuss the difference between computer games and computer simulations:
    - **Games –** let you explore and do challenges in imaginary worlds, for fun
    - **Simulations –** pretend to copy a real-life situation (eg. flight simulator, virtual pet)
- Talk about how games have developed during the past few decades, asking questions like:
    - What is the earliest game/simulation you remember playing?
    - What was the challenge in it?
    - What did you use to interact with it?
    - Can you think of any games or simulations that were around before you were born?

Step 2

- Show a range of **input devices** on the IWB that are, and have been in the past, used for controlling games consoles (Resource6.3.1)
- Talk through each one and draw attention to how they have developed to allow players to interact with computer games much more easily, and naturally, over time
- Talk about how they have moved from joysticks, to touch screens and then on to wireless motion sensors – such as Wii/Wii U and now Xbox Kinect that allows the player to use their whole body as a controller



Core

Step 3

- Hand out Worksheet6.3.1 or load on computers for children to complete
- The children work in pairs to conduct internet research for each console and to create a short timeline of computer game control
- They give examples of input devices (controllers, joysticks, gamepads and wireless motion sensors) used for each example
- They can import (using internet image searches or images available within the Resource6.3.1 folder) pictures or draw screenshots of example games and corresponding controllers

Differentiation

Easier: Children could use a shorter timeline

Harder: Children could write about how they think each input device works (eg. wireless sensor – how does the information get transferred from the input device to the console for processing?)

Extension

The children could design a futuristic controller themselves. How would the player interact with the controller? What would it do?

Resources: iPads; Pyonkee app; Resource 6.3.2a; Resource 6.3.2b; Program6.3.2.sb

Objectives

- To program a computer game by sequencing conditional statements

Success Criteria

- The children's program a simple 'pong' style computer game using **if...then** programming structures

Vocabulary

control; input; output; process; condition; statement; if; then

Step 1

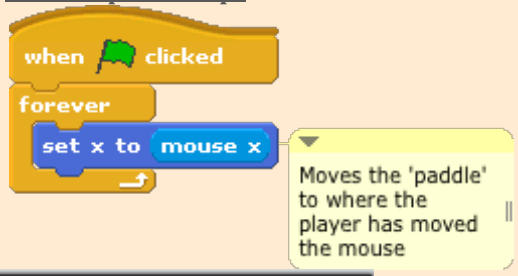
- Remind the children of their work in the last session where we explored various computer games and their input devices (controllers)
- Ask – *what is computer control?*
  - An **input** device transfers information from the outside world into the computer
  - The computer then **processes** the information that has been input
  - The computer responds by producing **output**
- Show a few computer games images on the IWB (Resource 6.3.2a) {eg. Etch-a-Sketch; Pong; Angry Birds} (Resources)
- Ask :
  - *how does the player control these games?*
  - *Which **conditional** (if..then) **statements** do you think would be in them?*
  - *eg.* **if** right-arrow-key-pressed
    turn-right

- Etch-a-Sketch draws shapes by the player turning knobs on the console
- Pong simulates table tennis with the player vertically moving an on-screen 'paddle' by turning the knobs on a controller to hit an on-screen ball
- Angry Birds, typically, fires birds at on-screen pigs using touch screen technology

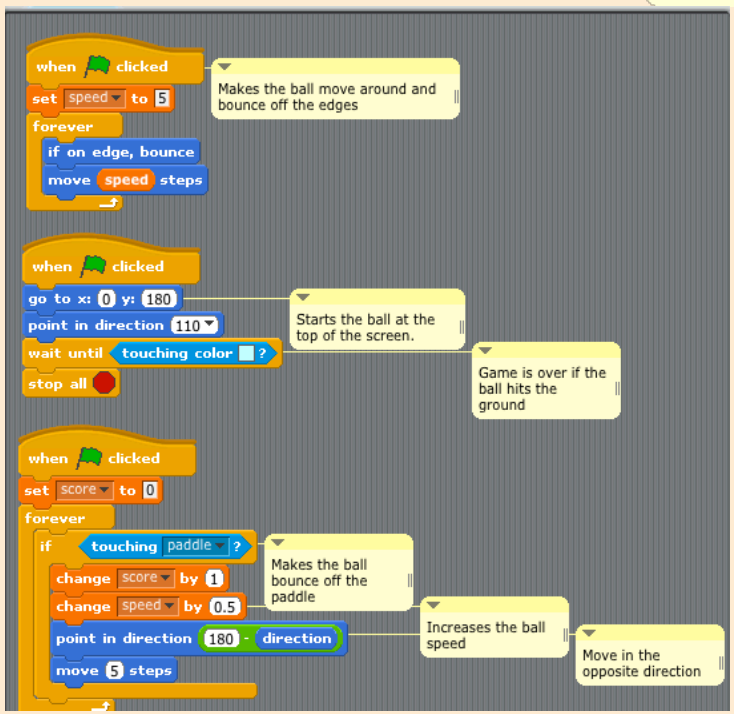
Step 2

- Show a Pyonkee pong game on the IWB and show the children how to play it (Program6.3.2.sb)
- Without showing the script blocks, ask the children which ***conditional statements*** (**if on edge; if touching**) do they think the game might be using?
- Quickly remind the children of the main elements of the interface
- Show the children how to:
  - Select command blocks from the different block pallets
  - Snap blocks together that will **execute** in **sequence**
  - Delete command blocks eg. by right tapping & flicking to the left
  - Use commands to reset the stage area (eg. clear)
  - Use **point in direction** blocks
- Demonstrate which moving sprites (objects) will need to be created to program a game with the same functionality and how to rename them
- Model how to design a background using the Paint Editor
- Create some **variables** for speed and score

Paddle sprite script



Ball sprite script



Core

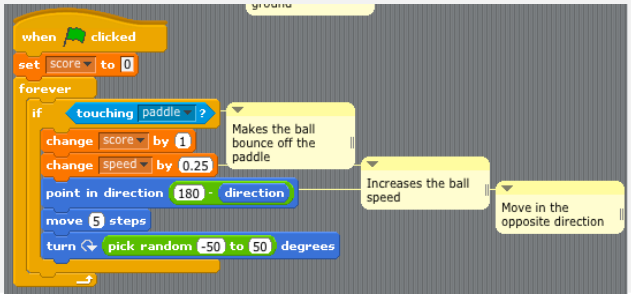
Step 3

- Children use Resource6.3.2 for support to program their own 'Pong' style computer game
- They use conditional **if** statements, **sensing** blocks and **variables** to make an object bounce off another and increase a score when it does

Differentiation

Easier: Some children could use an existing project where some script blocks have already been programmed

Harder: Some children could make the ball movement more random by using the turn motion block and random operation block. Challenge them to try to increase the randomness of movement by experimenting with the range of values



Extension

Challenge the children to program the ball to leave a trace of its movement by using the pen tool

iProgram 6.3.3 iPlan Year 6

Resources: iPads; Pyonkee app; Program6.2.3.sb; Resource6.2.3; Worksheet6.2.3 (enlarged to A3 or loaded on computers);

Objectives

- To understand that the behaviour of a computer program should be planned
- To understand that programs are developed according to a plan

Success Criteria

- The children record the sprites, backgrounds, interactions and scoring rules for an 'Angry Birds' style game

Vocabulary

Design; plan; logical operators (greater than, less than, equal to, less than and equal to, greater than and equal to); variables

Step 1

- Load a simplified Angry Birds style project onto the IWB (Program6.3.3.sb)
- Play it and discuss the main game elements. Explain that the player **inputs** the angle (from zero on the horizontal) to fly at in order to hit the pig
- Ask:
  - *what are the sprites?*
  - *How are the sprites **interacting** (eg. Bird touches the pig and scores a point)*
  - *their costume changes?*
  - *the sounds?*
  - *What is the background?*
  - *What are the scores/levels?*
- Model starting to complete a planning sheet (Worksheet6.3.3) on the board for the game

Core

Step 2

- The children complete Worksheet6.3.3 with their plans for their own Angry Birds style computer game
- Their game must involve:
  - A 'goodie' sprite
  - A 'baddie' sprite
  - The 'goodie' must hit the 'baddie'
  - The 'goodie' must win points
  - The game must have at least two levels

Differentiation

Easier: Some children may require help in expressing/developing/recording their ideas

Harder: Some children may be encouraged to increase the complexity of the scoring rules and include more levels in their games

Extension

Children swap their plans with partner. The other child should read the plan and explain, using it, how they think their partner's game will work. If the plans are not clear enough, the plans must be adapted. **NB**: explain that game **developers** often work from the plans of game **designers**

Resources: iPads; Pyonkee app; Resource6.3.3.sb; Worksheet6.3.3; Program6.3.3 assets shared with pupil iPads

Objectives

- To program an algorithm according to a plan

Success Criteria

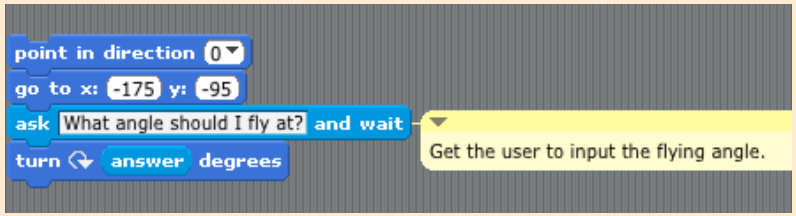
- The children create the sprites necessary for their game
- The children program their sprites to move

Vocabulary

algorithm; plan; sprite; costume; variable; iteration (repeat, forever, while); test; bug

Step 1

- Discuss last session's planning and show some good examples to the class, drawing attention to well named sprites, variable names and clear descriptions of the sprites' interactions
- Tell the children that, today, they will be using their plans to begin to put the code together for their computer games
- They start by importing/editing or drawing the sprites needed for their game and adding the blocks of code required for movement
- Encourage the children to test their code as they go along (systematically) to help them pick up any **bugs** as they are coded
- Tell them that testing in a systematic manner is what real programmers do and makes it easier to find out what is wrong with their code



Core

Differentiation

Step 2

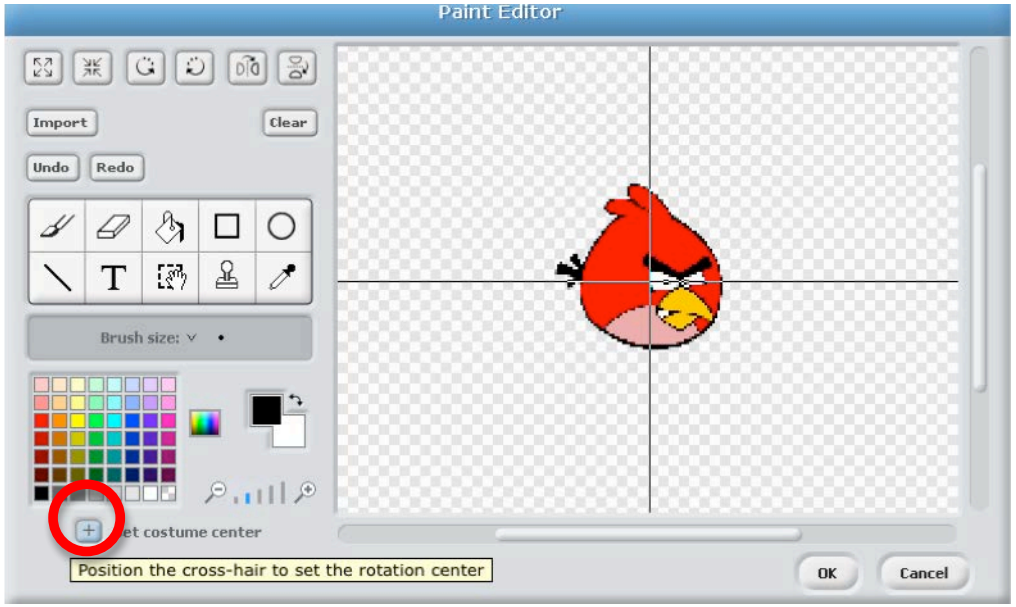
- The children use their plans (Worksheet6.3.3) from the last session to import/edit or draw the sprites needed for their game
- They import/edit the required costume changes for their sprites
- They then program them to move according to plan

Easier: Children could import ready-made sprites contained within the assets folder (resources)

Harder: Some children could be encouraged to add more 'baddies' the higher the score becomes and will therefore need to have more sprites and costume changes

Tip:

Remind the children to set their sprite's costume center in Paint Editor in order for the sprite to turn effectively



iPad 6.3.5 iDevelop

Year 6

Resources: Pyonkee app; iPads; Resource6.3.3.sb; Resource6.3.3; Worksheet6.3.3; Program6.3.3 assets

Note:

This session may require more than one session

Objectives

- To develop a program according to a plan

Success Criteria

- The children add variables and program the scoring rules necessary to progress in their game
- The children add the backgrounds and background changes for level changes in their game

Vocabulary

Algorithm; plan; sprite; costume variable; iteration (repeat, forever, wait)

Core

Step 1

- The children continue to develop their computer games according to their plan
- They import/design at least two background costumes for 2 different levels
- They add and name variables necessary for the scoring and leveling rules of their game
- They award points for their 'goodie' sprite touching the 'baddie' and change levels (background costume) when a certain number of points are achieved
- Encourage the children to test their code as they go along (systematically) to help them pick up any **bugs** as they are coded which makes it easier than going through each line of code at the end

Differentiation

Easier

Some children may need help understanding the need for variables and where to use them in their code

iPad 6.2.6 iDebug Year 6

Resources: iPads; Pyonkee app; Worksheet6.3.6; Children's project files; Resource6.3.3

Objectives

- To develop strategies for testing and debugging computer programs

Success Criteria

- The children can identify mistakes in their own and other's computer programs
- The children make changes to their code when things do not execute as expected

Vocabulary

test; bugs; debug; amend; systematically

Step 1

- Ask the children if they have any problems with their games?
- Ask the children to discuss in pairs:
  1. What was the problem?
  2. How did you identify it?
  3. How was it fixed?
- Tell the children that real programmers test their code systematically. They have a testing plan which makes sure That they go through all possible ways of executing their code making sure that each statement is executed They deliberately try to make the code fail in order to make sure that the code is **bug** free

Core

Step 2

- The children play and test their games systematically to make sure they work as expected
- When they encounter problems with their programs they analyse their code and debug it
- They then swap places with another pupil and test the other pupil's game
- They record any **bugs** they encounter (Worksheet6.3.6)
- When complete, they discuss their findings
- The children then return to their own program and correct any identified bugs

Differentiation

Easier

Some children could benefit from adult support in identifying and fixing problems

Extension

The children can evaluate each other's work and discuss how they think their projects could be extended or improved

Assessment

Record of progress Write names in the appropriate box, with jottings on children on children whose attainment differs markedly from their group.	Expectations What children know, understand and can do
Some children will have not made as much progress and will:	• Write or amend computer programs to produce specific actions with assistance • Know that commands can be given in shorter form • Use iteration (repeats and loops) with assistance
Most children will: 	• Write and amend more complex computer programs to create a variety of outcomes • Use iteration(repeats and loops), variables and conditional statements (if..then) in computer programs • Test computer programs and correct most errors
Some children will have progressed further and will: 	• Create procedures that call on other procedures using broadcasting blocks and produce a result • Create and use efficient methods of iteration, and nested conditional statements (if..then..if etc.) • Systematically test computer programs for bugs and make them work as expected • Critically analyse code and suggest more elegant solutions
Transition Guide - Working at Level 4	