Coding and Programming in Key Stage 2

Activity 1 – Scratch Conversations

Computing National Curriculum areas covered (all Key Stage 2):

- design, write and debug programs that accomplish specific goals; solve problems by decomposing them into smaller parts
- use sequencing in programs; work with various forms of output
- use logical reasoning to detect and correct errors in algorithms and programs
- select, use and combine a variety of software to design and create programs that accomplish given goals, including presenting data and information

Task overview:

- Familiarisation with the Scratch interface
- Create a conversation between two characters using Say blocks

Activity cross-curricular links:

- History – interviewing historical figures, linked to a topic of your choice

Other possible cross-curricular links:

- Literacy – speaking and listening; drama; speech punctuation; feelings and emotions of characters
- Geography – conversation between two people from contrasting environments
- PSHCE – discussing feelings towards an issue, such as bullying
- Science – highlighting misconceptions about a particular topic
- MFL – write or record dialogue for the character in a different language
- Art and design – design characters and backgrounds using a range of techniques, including ICT
Activity 1 – Scratch Conversations

The Scratch environment has four screen sections:

- **Script Area** – contains your program’s code
- **Block Palette** – contains sections of code (called Blocks), which can be dragged into the Script Area
- **Stage** – where your program’s actions take place
- **Sprites Area** – contains details of the characters (called Sprites) in your program
Our first task is to create a conversation between two characters. Before working at the computer, you may wish to get the children to think about what their characters will say. In class, the techniques discussed within the other sessions, such as Computing Unplugged, should be used to introduce concepts.

<table>
<thead>
<tr>
<th>Task explanation</th>
<th>Required steps</th>
<th>Screenshots</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firstly, delete the cat sprite so we can use something more interesting</td>
<td>Click with the right mouse button on the cat in the Sprites Area and select Delete</td>
<td><img src="image" alt="Delete Button" /></td>
</tr>
<tr>
<td>Next, we’re going to choose our characters</td>
<td>Click on the Choose new sprite from library button at the top of the Sprites Area&lt;br&gt;&lt;br&gt;Select a sprite from either the Animals, Fantasy or People sections and press OK&lt;br&gt;&lt;br&gt;Repeat to add a second Sprite</td>
<td><img src="image" alt="Choose New Sprite" /></td>
</tr>
<tr>
<td>We now need a more exciting background</td>
<td>Click on the Stage (left-hand side of the Sprites Area)&lt;br&gt;&lt;br&gt;Click on the Choose backdrop from library button&lt;br&gt;&lt;br&gt;Select a suitable background</td>
<td><img src="image" alt="Choose Backdrop" /></td>
</tr>
<tr>
<td>It’s now time for our sprites to have a conversation. To do this, we need a Green flag to start the program (Script) and a number of instructions in order</td>
<td>Click on your first Sprite in the Sprites Area. Ensure you can see the required blocks by clicking on Scripts at the top of the Block Pallet&lt;br&gt;&lt;br&gt;At the top of the Block Pallet, click on Events and drag a when Green Flag clicked block into the Script Area&lt;br&gt;&lt;br&gt;Drag two Say blocks from Looks into the Script Area and attach them to the when Green Flag clicked block&lt;br&gt;&lt;br&gt;Enter some text for the sprite to say&lt;br&gt;&lt;br&gt;Click on your other Sprite and repeat the above</td>
<td><img src="image" alt="Sprite 1" /> <img src="image" alt="Sprite 2" /></td>
</tr>
</tbody>
</table>
Click on the Green Flag at the top of the Stage to run your program

You will have noticed the Sprites talk over each other! To rectify this problem, we need to use a \emph{Wait} instruction.

Click on your first Sprite and drag two \emph{Wait} blocks from Control into your existing \emph{Script}.

Change the \emph{Wait} time to two seconds for each block.

Click on your other Sprite and repeat the process (note the location of the \emph{Wait} blocks within the \emph{script} is different for the two Sprites).

Run your program again using the Green Flag.

Congratulations! You’ve created your first program in Scratch. You’ve written a program to achieve a specific goal, sequenced instructions and worked with outputs (the speech displayed on the screen). You’ve also probably corrected errors in your program, which is known as \emph{debugging}.

\textbf{Extension activities}

For each of the activities below, feel free to experiment with a range of features to make your program as engaging as possible.

- Get the Sprite to move towards the character (Hint: have a look in \emph{Motion}).
- Record and play a sound instead of using a \emph{Say} block (Hint: try looking in \emph{Sound}).
- Change the appearance of the Sprite after they’ve spoken (Hint: use a \emph{Next Costume} block in \emph{Looks}).
- Alter the background after one of the characters has spoken (Hint: the required block is in \emph{Looks}).
- Add a third character (Sprite) to the conversation.
- Draw your own character (Sprite) or search for suitable sprites online.
Coding and Programming in Key Stage 2

Activity 2 – Maths Quiz

Computing National Curriculum areas covered (all Key Stage 2):

- 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 logical reasoning to detect and correct errors in algorithms and programs
- select, use and combine a variety of software to design and create programs that accomplish given goals, including collecting, analysing and presenting data and information

Task overview:

- Create a Maths game containing a range of multiplication questions
- Add a score to the game using a variable – optional

Activity cross-curricular links:

- Mathematics – division and multiplication facts

Possible cross-curricular links:

- All subjects – assessment activities (formative and summative; peer assessment)
- Mathematics – other subject areas, including Conversions between units of measure; Comparing decimals, percentages and fractions
- Science – predicting outcomes of experiments
**Activity 2 – Maths Quiz**

Initially, we need to ask the user a question and allow them to **Input** the answer. Although quizzes can be produced for any topic, it’s best to initially only ask questions with numerical answers, which minimises errors relating to spelling or typing.

<table>
<thead>
<tr>
<th>Task explanation</th>
<th>Required steps</th>
<th>Screenshots</th>
</tr>
</thead>
<tbody>
<tr>
<td>As with the first task, delete the cat <em>Sprite</em> so we can use something else</td>
<td>Click with the right mouse button on the cat in the <em>Sprites Area</em> and select <em>Delete</em></td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
</tbody>
</table>
| Next, we’re going to choose our character (you only need one!)                   | Click on the *Choose new sprite from library* button at the top of the *Sprites Area*  
Select a sprite from either the *Animals, Fantasy or People* sections and press *OK*  
Feel free to also change the background | ![Image](image2.png) |
| We need our character to introduce the game when the program starts             | Drag a *when Green Flag clicked* block and a *Say* block into the *Script Area* (if you can’t see the *Script Area*, click on the *Scripts* tab at the top of the window)  
Change the text to introduce your game | ![Image](image3.png) |
| Our quiz needs a question to ask the user                                        | Drag an *Ask* block from *Sensing* and attach it to your *Script*  
Change the text to ask a multiplication question | ![Image](image4.png) |
| The program has to display a different message depending on whether the answer is correct or incorrect. We do this using *Selection* | Drag an *if, then, else* block from *Control* and attach it to your *Script* | ![Image](image5.png) |
Following this, we need to check the *Input* and use *Selection* to choose the route through our program, which is:

*If* the answer is correct (2x5=10), *then* display a “well done” message, *else* display an “unlucky” message

| Drag an *equals* block from *Operators* into the top of the *if, then, else* block |
| Place *Answer* from *Sensing* in one side of the *equals* block, followed by the correct answer on the other side |
| Drag a *Say* block into the first half of the *if, then, else* block. Change the text to “Well done” |
| Drag another *Say* block into the second half of the *if, then, else* block. Change the text to “Unlucky” |
| Run your program using the *Green Flag* (you enter your answer in the box that appears at the bottom of the *Stage*) |

| Add some additional questions using the same process |
| Add further *Ask* and *if, then, else* blocks to your *Script* containing different questions and answers |
| Run your program and check it behaves as expected |

| Add some additional questions using the same process |
| Add further *Ask* and *if, then, else* blocks to your *Script* containing different questions and answers |
| Run your program and check it behaves as expected |

Well done! You’ve made another program in Scratch. This time you’ve also used *selection* and worked with both *inputs* and *outputs*.

Depending on your confidence, you can now either try the extension activities below, or try to implement a *variable* to record the score, as shown overleaf

**Extension activities**

For each of the activities below, feel free to experiment with a range of features to make your program as engaging as possible.

- Make the character move or dance when an answer is correct (Hint: look in *Motion*)
- Get the character to include the user’s answer when they speak – e.g. "Well done. The answer was 10" (Hint: you’ll need to use a *Join* block from *Operators*)
- Add repetition to your program by returning to the start once all the questions have been answered (Hint: use a *Forever* loop in *Control*)
- Create a "Game over" screen (background containing no characters) once the quiz has finished
Optional activity – Using variables

We’re now going to add a score to our game using a variable. A variable is similar to a box. It can contain anything the computer can store, such as numbers or text. Its contents can be changed, or varied (hence the name variable), and we can find out the contents of the box at any time.

<table>
<thead>
<tr>
<th>We first need to create a variable to store the score</th>
<th>Click on Data at the top of the Block Palette and select Make a Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Give the variable a name, such as “Score” and click OK</td>
</tr>
<tr>
<td></td>
<td>If the tick box next to the variable name is selected in the Block Palette, the variable is shown in the corner of the Stage</td>
</tr>
<tr>
<td>When the quiz starts, we have to ensure the score starts at zero, rather than continuing from the previous game</td>
<td>Drag the Set, to block from Data to immediately below the when Green Flag clicked block in your script (this ensures it’s the first instruction run)</td>
</tr>
<tr>
<td>Every time the user gets an answer correct, we need to increase the score by one</td>
<td>Drag a Change, by block from Data into the first half of your if, then, else blocks. Place it before the “Well done” message</td>
</tr>
<tr>
<td></td>
<td>Repeat the process for the other if, then, else blocks in your Script</td>
</tr>
<tr>
<td></td>
<td>Run your program and check the score starts at zero and it increases by one when a question is answered correctly</td>
</tr>
</tbody>
</table>

Good work! You’ve now used a variable to record the score. You have also output the variable to the screen for the user to see.

Extension activities

For each of the activities below, feel free to experiment with a range of features to make your program as engaging as possible.

- Adjust the score to decrease when the user gets a question wrong (Hint: change the score by -1)
- Change the background when the score reaches ten (Hint: add an if, then block to check the score and perform an action if Score = 10)