Brough Primary School – Curriculum Intention Plan 2021 - 2022

Subject: Computing **Year Group:** Year 3/4		**Area of learning:** We are Software Developers Rising Stars 4.1
Links to previous work/Remember when	Scratch	
Term	**Year**	**Key Skills to be taught**
Summer 1 ***(Cycle B)*** **2022** What the children should know at the end of this series of lessons	Y3/4	The pupils start by playing and analysing educational computer games, identifying those features that make a game successful. They then plan and design a game, with a clear target audience in mind. They create a working prototype, and then develop it further to add functionality and improve the user interface. They test their game and make any necessary changes.

Vocabulary:

interactive, input, output, sequence, debug, algorithms, detect, selection, repetition, variables

Sequence of learning	Objectives and suggested details provided by the subject leader.
1	**Step 1: Playing and analysing educational games** Give the pupils time to play some educational games (see Resources), and then ask them to describe the algorithms behind them. (These are likely to involve question, response and feedback, but might include other elements that pupils may be able to discover by trying different answers or otherwise changing how they interact with the program.) Discuss how many of the games are based on the repeating pattern of question/response/feedback. Ask the pupils to identify what sets good games apart. Try to elicit answers that refer to progression, challenge, interaction and context. Tell the pupils that they will be creating an educational game in Scratch. Ask them to think about (or brief them on) what age group their game is for, the topic they will cover, and the questions they'll set. Ask them to think about how they will manage interaction, i.e. what forms of input and output they'll use. **Suggest resources** www.learninggamesforkids.com www.topmarks.co.uk/maths-games
2	**Step 2: Building a game prototype** Ask the pupils to remind one another how the Scratch interface works, and explain what the blocks they've used already do. If pupils are not familiar with Scratch, spend some time showing them how it works (see Unit 3.1 – We are programmers and associated walkthroughs).

Brough Primary School – Curriculum Intention Plan 2021 - 2022

	Challenge the pupils to develop a simple program in Scratch that asks a question and provides feedback depending on whether the answer is right or wrong. (For examples of Scratch programs that follow this structure, see My Rising Stars.) Depending on how familiar the pupils are with Scratch, you might need to demonstrate how the if/then/else selection block works, or perhaps get one or more of the class to show this to others. (See the first weblink in Resources.) Encourage the pupils to test their programs and to debug them if necessary so that the right/wrong feedback is given correctly for right/wrong answers. You will probably need to give pupils a lot of help with debugging. Take feedback on common errors the children made when programming to share as a class, inviting other pupils to provide advice or perhaps giving it yourself. It is worth spending some time explaining the idea of a variable to the pupils, either now or later, when they've had some experience of working with them. A variable lets computer programs store, retrieve or change simple data – a variable is typically thought of as a particular bit of the computer's memory that holds a specific bit of data. Show the pupils how the random number block can be used to change the numbers used in arithmetic questions, and how this can be combined with variables to allow the computer to work out the correct answer for the random questions asked. (See the second weblink in Resources.) Show the pupils how the join block allows variables to be used in questions. **Suggested resources** **https://en.scratch-wiki.info/wiki/If_()_Then,_Else_(block)** **https://en.scratch-wiki.info/wiki/Pick_Random_()_to_()_(block)**
3	**Step 3: Adding in repetition and keeping track** Discuss the pupils' ideas about common features of educational games, focusing on repetition of questions. Demonstrate (or allow pupils to demonstrate) how a repeat loop can be used to ask a number of different random questions. Show pupils how the repeat loop could be used to stop the game when pupils have been asked a certain number of questions (say ten), or perhaps how the game could end when they've got a certain score or when a countdown timer reaches zero. Ask the pupils to add repetition into their games, testing and debugging them to ensure they still function correctly. Ask them to add a 'game over' message when the correct number of questions has been asked. Remind the pupils about variables and show (or allow pupils to show) how the computer can use a variable to keep track of the score for the game, perhaps as the number of correct answers or by calculating the percentage correct so far. Ask the pupils to add some way of keeping score to their game, again testing and debugging their games to ensure they still work correctly. Ask the pupils to add in a message at the end of the game to say what the player's final score was. It is unlikely that the pupils' scripts will work at first. That's fine: it's by fixing things that we often learn how they work. When children do encounter difficulties, spend time getting them to explain, as logically as they can, how their algorithm should work, and to use this reasoning to work out what they need to do to debug their scripts. Encourage them to experiment to see what effect the changes they make have.
4	**Step 4: Working on the interface** Remind the pupils of what they noticed about the educational games they played at the beginning of the unit. Use questioning to get them to think about the elements of interface design that made games more appealing and engaging. They should, with your help, be able to identify graphics, sound and interaction as key aspects of this. Ask the pupils to work on the graphics elements of their games, focusing particularly on the sprite that asks questions, but perhaps also thinking about how other graphics might be used to measure progress in the

	game. Ask the pupils to work on the interaction in their games, thinking about how they could improve the way the computer responds if they get an answer right or wrong and how their progress can be shown, perhaps by showing the sprite moving towards a goal or by having the sprite eat from a reducing pile of apples, or something similar. Ask the pupils to look at adding sound effects or voice prompts to their games. If pupils have not used sound recording in Scratch before, demonstrate how to use the Sounds tab to record dialogue or sound effects using a microphone, and how to use the play sound until done block in the Sound palette under the Code tab to play the sound back. Tell or remind pupils that recordings can be given names, to make it easier to remember which is which. (A small number of sound effects are pre-loaded with Scratch.) Again, ask the pupils to thoroughly test and debug their games.
5	**Step 5: Building in progression** Ask the pupils to compare the educational games they looked at earlier with other games that they are familiar with, such as Dots, Candy Crush or the Angry Birds series. What differences do they notice? Can they see ways in which their games, and educational games in general, might be made more engaging? They should identify the idea of levels or progression as one common characteristic of commercial games. Ask them to suggest ways in which they could build in additional levels or progressively increasing difficulty to their games. Demonstrate one way in which they might do this, perhaps through using a further variable to determine the size of the random numbers used in an arithmetic question, or by having another set of questions, with larger numbers, when the first level of the game is completed. Another, more advanced, possibility is to use variables (or lists) to remember questions that pupils got wrong and to ask those questions again. Give the pupils time to add in at least one additional level, with harder questions, to their game. Remind them that they need to test and debug their games thoroughly.
6	**Step 6: Testing and refining** Remind the pupils of the criteria they identified in Step 1 for successful educational games. Ask them to review their own games using this list. Provide an opportunity for the pupils to test and review one another's games, checking for any bugs that remain as well as using the criteria identified earlier. If the pupils have been developing games for a different age group, it would be useful to let children in the target audience play the games and provide feedback to the developers. Provide time for the pupils to refine their games in the light of the feedback they've received. Use some of this time to ask the pupils to explain to you how their games work. Give the pupils a chance to present their finished games to the class, perhaps in the style of a Dragons' Den-style pitch in which they make a case for their game to be developed further or turned into an iPad app or similar. In making their pitch, the pupils could draw on the technical ideas incorporated into their game, the learning objectives their game addresses, and the feedback from their users. The children should evaluate the success of their work.

Learning Outcome/product

Brough Primary School – Curriculum Intention Plan 2021 - 2022

Assessment records	List only those children who have not achieved the expected outcomes
	This unit will enable the children to: develop an educational computer game using selection and repetition understand and use variables start to debug computer programs recognise the importance of user interface design, including consideration of input and output.
	Children working above.

End of unit assessment question
What is an example of an educational interactive game? How do you debug a programme? How do you create a sequence?