|  |  |  |
| --- | --- | --- |
| **Coding games** | | |
|  |  |  |
| Design and make your own computer game | | |
| **Subject(s):** Computing  **Approx time:** 70 - 120 mins (depending on prior experience with Scratch) |  | **Key words / Topics:**   * Coding * Variables * Selection * Gaming * Computing |
| **Stay safe**  Whether you are a scientist researching a new medicine or an engineer solving climate change, safety always comes first. An adult must always be around and supervising when doing this activity. You are responsible for:  • ensuring that any equipment used for this activity is in good working condition  • behaving sensibly and following any safety instructions so as not to hurt or injure yourself or others  Please note that in the absence of any negligence or other breach of duty by us, this activity is carried out at your own risk. It is important to take extra care at the stages marked with this symbol: ⚠ | | |
| **Suggested learning outcomes** |  |  |
| * To be able to identify some influential coders * To use selections and variables when coding * To be able to create a game with a score and timer * To use flowcharts to create and debug code | | |
| **Introduction** |  |  |
| This is one of a set of resources designed to allow learners to develop their knowledge and skills in Computing and Art and Design. This resource has been developed with assistance from Archives of IT and focusses on coding. digital art. The main activity involves learners planning and coding a game in Scratch, using selections and variables. It also includes a brief history of how coding has developed and notes the works of influential coders.  This lesson focuses on exploratory learning, giving the students time to explore and work out for themselves how to create the code needed for their game. | | |
| **Purpose of this activity**  In this activity learners will develop their ability to use coding.  This activity could be used as a main lesson activity, to introduce the concept of coding. Alternatively it could be used to introduce or reinforce how to use block coding with Scratch. | | |
|  |  |  |
| **Activity** |  | **Teacher notes** |
| **Introduction (10-15 minutes)**  Teacher to explain what coding is and that learners are going to plan and code a functioning game with selections and variables that they will then have an opportunity to play in the class. Teacher to explain how coding has changed over the years, giving mention to inspirational coders Margaret Hamilton, Grace Hopper and Mark Zuckerberg.  **Variables (10-20 minutes)**  Ask selected learners to have a go at playing ‘whack a mole’ on slide 9 of the presentation. Ask learners to think about how the score and timer works.  Introduce variables and ask the learners to write a number on one sticky note and an animal on another. Ask different learners to come up to the board and add in their sticky notes, replacing the previous notes.  **Designing the game (10-15 minutes)**  Teacher to hand out the coding games planning sheet for learners to complete as they plan their games..  **Making their game (40-60 minutes)**  Learners should explore the tools available on scratch to see if they can work out which code is needed for their game. Learners should make a note of their flowchart, noting what worked and what needs to be changed.  Learners should add a timer into their game so it does not go on forever. |  | Learners will need to create an account to save their games on Scratch. This is free to do, however it does require an email. The teacher may therefore want to create a class account or individual accounts for the learners before the lesson.  **Variables**  The game has an ad at the start so worth loading before the lesson starts.  **Designing the game**  Learners may wish to open scratch when planning their game to see all of their character and sound options.  Learners should start with three characters but can always add in more later once they are sure that their code is correct  **Making their game**  There are step-by-step instructions in the video for how to create code using Scratch.  Remind learners:   * to use as little code as possible by using the repeat blocks. * the wait and show times are important. If there is an animal that gives a big score, learners could make the show wait time shorter so that it is harder to get. * they will need two separate flow charts for each character. * They should test their code frequently, by clicking on the green flag. * They should save their work frequently. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Differentiation** |  | |  |
| **Basic** |  | | **Extension** |
| * Teacher to provide a pre-coded game that children can edit/amend. |  | | * Add in new screens that the game goes to, to indicate if the player has won or lost. * Ask the children to move round the class and have a go at the different games. Allow the students to take post-it notes so they can leave 2 stars (things that they like about the game) and a wish (something to improve). If time is available, learners can then go back and edit their work if needed. |
|  |  | |  |
| **Resources** |  | | **Required files** icon-docicon-pdficon-ppt |
| * Computers / laptops / tablets with internet access. * Sticky notes. |  | | icon-ppt Presentation – Coding games  icon-doc Coding games planning sheet |
|  |  | |  |
| **Additional websites** | | | |
| * Video story of Nasa’s first software engineer – Margaret Hamilton: https://www.youtube.com/watch?v=kYCZPXSVvOQ * Scratch – Free coding community for kids <https://scratch.mit.edu> * Whack ‘Em All – Example game to provide inspiration <https://www.classicgame.co.uk/game/Whack+a+Mole> | | | |
|  |  | |  |
| **Related activities (to build a full lesson)** | | | |
| **Starters** (Options)   * Complete an acrostic for Coding, using slides 2 and 3 in the presentation. | | **Plenary**   * Learners share their final games with the class for comment/approval. Discuss what went well and how they could improve/extend their games. | |

|  |
| --- |
| **The Engineering Context** |
| * Coding is an essential part of software engineering – all automated devices, systems, apps and any device with a microcontroller are controlled by algorithms. |

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |  |
| --- | --- |
| **Curriculum links** | |
| **England: National Curriculum**  KS2 Computing   * 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 technology safely, respectfully and responsibly; recognise acceptable/unacceptable behaviour | **Northern Ireland: Curriculum**  KS2 Mathematics and Numeracy  Shape and space / position, movement and direction   * … be introduced to a programming language and use it to create pictures and patterns and to generate shapes |

|  |  |
| --- | --- |
| **Scotland: Curriculum for Excellence**  Technologies  Computing Science   * TCH 2-13a: understand the operation of a process and its outcome. I can structure related items of information. * TCH 2-14a: I can explain core programming language concepts in appropriate technical language. * TCH 2-15a: I can create, develop and evaluate computing solutions in response to a design challenge. | **Wales: National Curriculum**   * Cross-curricular skills: digital competence   Computation is the foundation of our digital world  Progression step 2   * I can safely use a range of tools, materials and equipment to construct for a variety of reasons. * I can use computational thinking techniques, through unplugged or offline activities. * I can create simple algorithms and am beginning to explain errors. * I can follow algorithms to determine their purpose and predict outcomes. |
|  |  |

|  |
| --- |
| **Assessment opportunities** |
| * Formal summative assessment of completed work by the teacher. * Peer assessment and feedback on coding games produced. |