Tackling the Technology Gender Gap Together:

Interactive Story Writing

Level - Third Level

Subject area/s – Computing Science and Literacy

Interactive Story Writing

Level - S1/S2 Third Level

Subject area/s – Technologies - Computing Science, Languages

Experiences and Outcomes –

ENG 3-31a:

Having explored the elements which writers use, I can create texts in different genres by:

- integrating the conventions of my chosen genre successfully and/or
- using convincing and appropriate structures and/or
- creating interesting and convincing characters and/or
- building convincing settings which come to life.

TCH 3-15a: I can select appropriate development tools to design, build, evaluate and refine computing solutions based on requirements.

Benchmarks - Technologies – Computing Science Third Level

- Designs and builds a program using a visual language combining constructs and using multiple variables.
- Can find and correct errors in program logic.
- Writes code which receives and responds to real world inputs (in a visual language).
- Groups related instructions into named subprograms (in a visual language).
- Writes code in which there is communication between parallel processes (in a visual language).

Prior Learning - Computing Science Second Level Benchmarks

Pupils should have prior understanding of genre and should be confident in story writing.

Pupils should be familiar with Scratch as a visual programming language. They should be able to create sprites and backdrops independently.

Pupils should have already met the relevant second level benchmarks before progressing on to this lesson. Pupils will be expected to have an understanding of conditions, repetition and variables. The following second level benchmarks will be necessary for pupils to progress in this lesson:

- Explains the meaning of individual instructions (including variables and conditional repetition) in a visual programming language
- Predicts what a complete program in a visual programming language will do when it runs, including how the properties of objects for example, position, direction and appearance change as the program runs through each instruction.
- Creates programs in a visual programming language including variables and conditional repetition.
- Identifies patterns in problem solving and reuses aspects of previous solutions appropriately for example, reuse code for a timer, score counter or controlling arrow keys.
- Identifies any mismatches between the task description and the programmed solution, and indicates how to fix them.

Duration of time – Approximately 11 x 50 minute lessons.

Computing Science Concepts and Approaches

Decomposition – breaking down game problem into smaller tasks such as sprite movement, interaction.

Debugging – testing game to ensure code works as expected finding bugs and fixing them when required.

Algorithms – writing algorithms to program the aspects of the game.

Overview of learning

Pupils will be given a story starter, they will work independently to plan and then write an imaginative story which will form the basis for the storyline of a Scratch game. They will draw upon prior learning at level 2 to develop a simple game.

The game should include:

- sprites and backdrops matching the setting of the story
- sprite movement on keypresses
- sprite interactions using conditional statements

Pupils will then learn how to extend this basic game to include complex conditional statements for sprite interactions. They will create a subprogram and utilise it more than once.

They will evaluate their game by working with a partner to ensure it represents the story they wrote and test it to find any bugs.

Pupil Objectives

- I can write a short story providing a clear description of the main character and the setting.
- I can develop a plan for creating an interactive game based on my story.
- I can use prior learning to code a game that includes:
    - sprite movement on keypresses
    - sprite interaction using conditional statements
    - sprites and backdrops created to match the setting of my story
- I can extend my code to include complex conditional statements for sprite interactions
- I can create and use multiple variables for different purposes in my game
- I can create and use a subprogram in my Scratch game
- I can debug my code.
- I can test and evaluate my game.

Resources required

Pupils can create an account on the Scratch website and projects can be saved there or alternatively the desktop Scratch application can be used.

Games design document for planning features of game included in pack.

Story planning document for planning characters, plot and setting provided in lesson pack.

Resources provided in teaching PowerPoint:

- Story starter example ideas
- Examples of code for each objective

- Criteria for final game evaluation
- Example extension tasks

Example game found here: https://scratch.mit.edu/projects/153093511/

Introduction

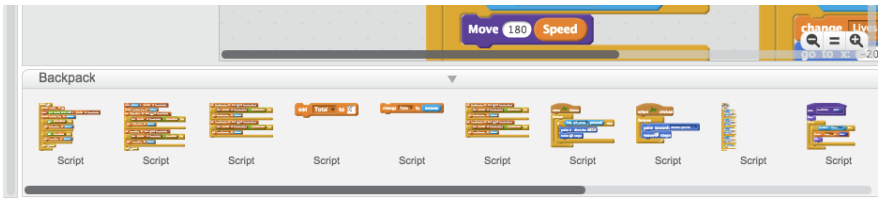
Introduce task to pupils and demonstrate example of a completed Scratch game. Over the course of 11 periods pupils will write a short story and then create a simple game based on it.

Teaching Notes

Point out to pupils the different block drawers in Scratch and how each one is a different colour. This will help them to find the blocks they are looking for.

It is helpful for pupils to see a demonstration of an example game being created alongside their own.

If you create a Scratch account online you can use the backpack feature to store chunks of code to be reused across different projects.



Example of game meeting all pupil objectives found here:
https://scratch.mit.edu/projects/153093511/

Main Activity

Lesson 1 – Story Planning (Slides 1-10)

Objective: I can plan a short story identifying the characteristics of my main character, story setting and plot.

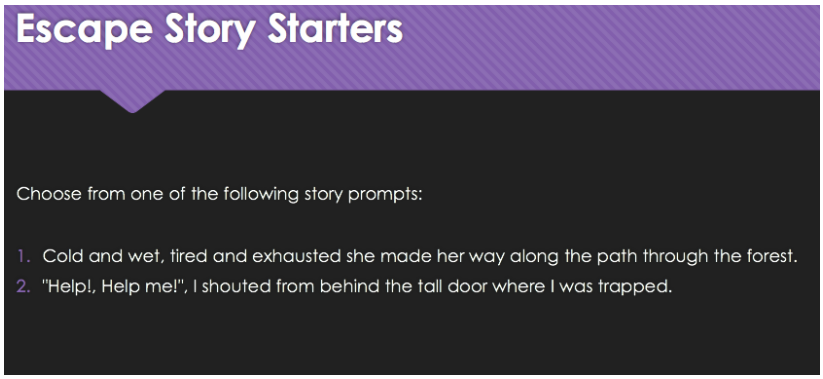
Introduce project to pupils, explaining what they will be learning about.

Story writing lessons 1-3 would ideally be delivered in collaboration with English department.

Pupils will write an imaginative short story based on one of the writing prompts provided in lesson PowerPoint.

Writing prompts can be changed to suit class, topic or type of game you would like pupils to be creating.

Theme suggested is “Escape”, this suits the type of maze style games that can be made in Scratch.



There is a story planning document provided in the lesson pack which asks pupils to think about the key characteristics of the main character, setting of the story and the plot. This plan should be completed during the first lesson.

At the end of the lesson, have pupils share their ideas for their story in small groups.

Questions include:

- What will your main character look like?
- What happens to your main character in the story?
- Where does the story take place?

Lesson 2 – Writing the story

Objective: I can write a short story clearly describing characters and setting based on a plan I have produced.

Pupils should now focus on writing their story using the planning document they completed in the last lesson.

Leave the story starters on the board as a prompt to remind them of the theme of their story.

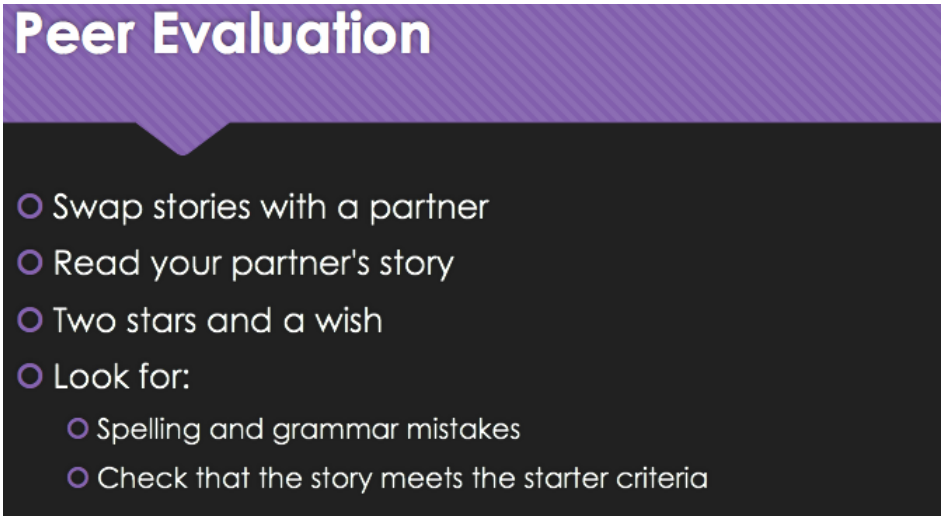
At the end of the lesson, give pupils time to stop and read over what they have written so far and to correct any spelling or grammar mistakes they can spot at this stage.

This lesson can be extended if more time is required to write the short story.

Lesson 3 – Evaluating the story (Slide 11)

Objective: I can work with a partner to share and evaluate each other's work.

Pupils should work with a partner to read each other's stories and evaluate them based on the criteria on slide 11 of the teaching PowerPoint.



The story should suit the chosen starter that has been used, it should also feature a main character that has been clearly described as well as a clear setting.

Pupils should use two stars and a wish method for identifying two things they liked or thought had been done well in their partner's story and one things that they think could be improved upon. They should also mark the work of their partner and correct spelling and grammar mistakes.

At the end of the lesson, pupils should swap their work back and have time to make corrections and improvements based on the suggestions of their peer.

Lesson 4 – Planning the game (Slides 12-14)

Objective: I can plan the key features of my game including sprites and gameplay based on the short story I have written.

Pupils will spend this lesson planning the features of the Scratch game, remind pupils that the game they are planning is an interactive version of their story so the game should feature characters from the story and be set in the same place.

Pupils may prefer to focus on implementing an aspect of their story as a game instead of trying to program the entire storyline.

Pupils should understand the type of features they can implement in a game from prior learning at 2nd level. Show pupils the example game: https://scratch.mit.edu/projects/153093511/

They should use the game planning document provided in lesson planning pack to identify what sprites and backdrops will be required and how they will look, the gameplay (can you score points, lose lives, how do you win the game).

Draw comparisons between the planning process for writing a story and planning for developing a game.

Pupils should be aiming to create a simple game with one level and room for the game to be extended later. Maze style games are most suited to this project.

At the end of the lesson, have pupils share their plan for their game in small groups or with the class.

Questions include:

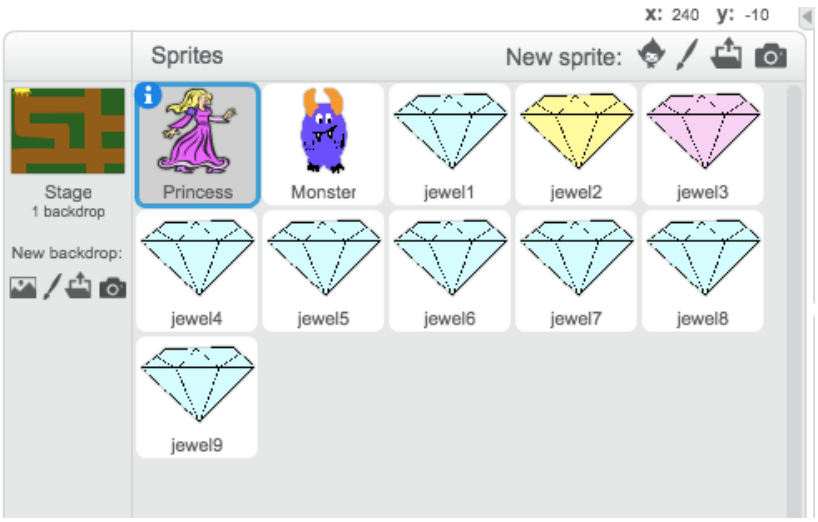
- How will you win the game?
- How many sprites will you need?
- Can you lose in the game?

Lesson 5 – Creating sprites and backdrops (Slides 15-17)

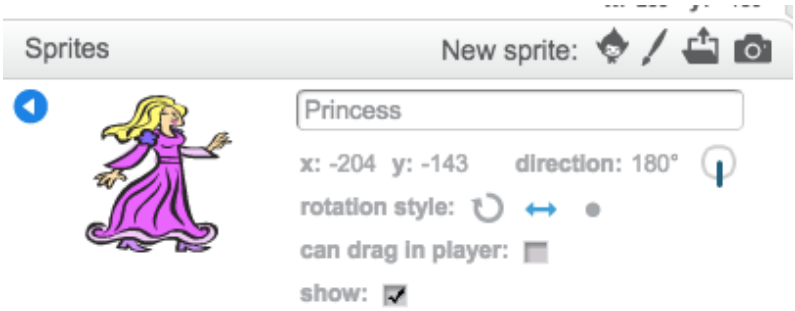
Objective: I can use the tools in Scratch to draw my own sprites and backdrops that match my story characters and setting.

Pupils will begin by creating an account on the Scratch website if they do note already have one: scratch.mit.edu

Pupils should be familiar with creating a new project and creating sprites in Scratch. They can then begin creating the sprites to match their plan. Sprites can be combination of new painted sprites and library sprites. Backdrops must be painted by pupils and link to imaginative writing piece.



Pupils should ensure they give their sprites meaningful names so that they can be easily identified. This will help them when they are writing code later in the project. This can be done by clicking on the small ***i*** in the top left hand corner of the sprite.



Lesson 6 – Programming sprite movement (Slides 18-20)

Objective: I can write code to make a sprite move on the screen.

Pupils should understand all concepts covered in Slides 18-20, as repetition and conditional statements are prior learning at 2nd level.

As a reminder, revisit algorithms on slide 18 and explain sequencing of instructions.

Show them the code on slides 19 and 20 and ask them to explain what is happening and what affect the code will have on a sprite.

Demonstrate what each movement example does using the program: https://scratch.mit.edu/projects/153093511/

Pupils should now implement suitable movement code for sprites in their game.

Lesson 7 – Programming sprite interactions and variables (Slides 21-25)

Objective: I can write code using variables and conditional statements to change scores or lives when a sprite touches another sprite in the game.

Explain the purpose of implementing sprite interactions using slide 21.

Revisit variables using slide 22, test what pupils remember from learning about variables at level 2. Ask what they have used variable for in the past.

Show examples of implementation of variables on slides 23-25. Ask pupils to identify what is happening in the code on each slide.

Show pupils an example of what each block of code will do in the sample program: https://scratch.mit.edu/projects/153093511/

Pupils should now implement suitable interactions and variables based on their plan for gameplay.

Lesson 8 – Programming complex conditional statements (Slides 26-29)

Objectives:

I understand the term complex condition

I can write code that uses a complex condition and operators in Scratch.

Revisit conditional statements using slide 26. Pupils should understand what the purpose of a conditional statement is and how it is used in a program from 2nd level learning.

They will have already added conditional statements into their game when implementing sprite interactions

Introduce the idea of a **complex** condition using slide 27, where the program must evaluate more than one statement before continuing.

In order to implement a complex condition we must use operators (slide 28). Scratch provides us with comparison operators (<,>,=) which will test whether the value on either side of the operator is greater than, less than or equal to the other. A comparison operator can be placed in either side of an **AND** or an **OR** block in order to determine if both conditions have to be true or if only one or the other must be true to proceed.

Ask class for examples of where we could use complex conditions in our games.

See winning example provided on slide 29 and see code demonstration using example game.

Pupils should now implement a complex condition in their game.

Lesson 9 – Programming subprograms (Slides 30-38)

Objectives:

I understand the differences in data types within a program.

I understand the terms subprogram and parameter.

I can implement a subprogram with parameters in my game.

Introduce the lesson by talking about types of data with pupils. Programming languages like to know what type to expect when you are inputting data. Scratch only deals with data of types; string, number and Boolean. Some programming languages have many more data types and usually have different data types for decimal and whole numbers.

Introduce the words string, number and Boolean to pupils using slides 30-31. Explain what each data type is, ask pupils for examples of the type of data they would hold in each type.

Number: whole number, decimal number. Examples: Age, Score.

String: any text or long number string. Examples: Name, Address, Postcode

Boolean: anything that can only have a true/false, yes/no value.

Introduce terms subprogram and parameter to pupils using slides 32-33 and explain purpose. See example of when to make use of a subprogram on slide 36.

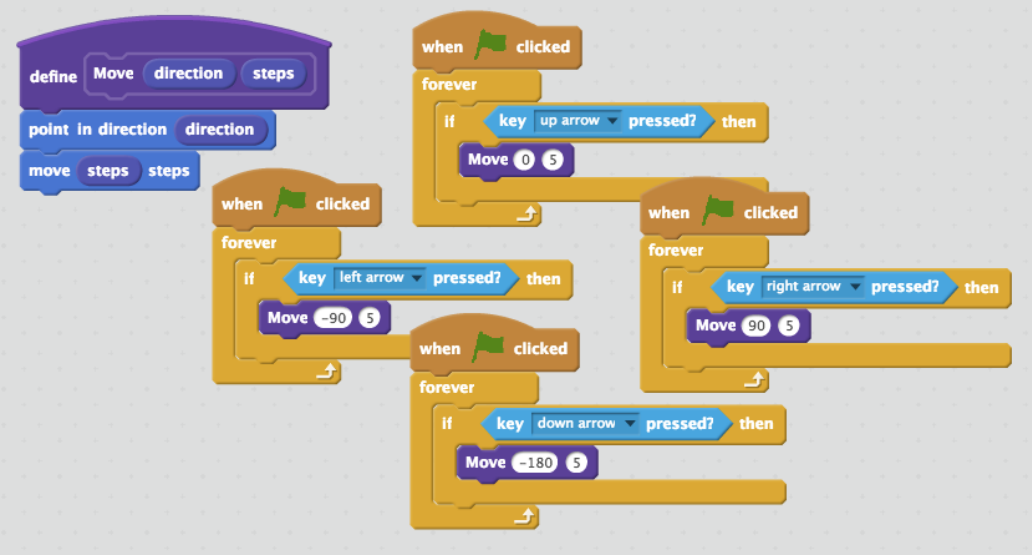
Talk through creating a subprogram with parameters with pupils using slides 37-38, implementing a subprogram and seeing it used in place of several blocks of repeated code will aid their understanding.

Ask them to extend this by using a variable as a parameter as seen on slide 39.

Pupils should now identify somewhere else in their program where they could implement a subprogram and do it independently.

Note: In Scratch if a subprogram block is created on one sprite it can only be used on that sprite, the subprogram would need to be recreated for use on any additional sprites.

Pupils may need additional time and support in this lesson to meet all objectives.



End lesson with exit pass asking pupils to identify the different data types and a reason for using subprograms.

Lesson 10 – Finalising and extending game (Slides 39-41)

Objective: I can ensure that my game meets all required criteria and extend its functionality if I have time.

Show pupils the success criteria for this project on slide 43. Allow pupils time to complete all requirements for the game. Pupils who have already completed required components can used this time to extend the game. Suggestions for game extensions provided on slide 41.

Pupils should ensure that game is playable and matches their plans as well as links closely with the original imaginative piece of writing.

Lesson 11 – Testing and evaluating Scratch game (Slide 42)

Objective:

I can test and debug my game.

I can evaluate my own game and the game of my partner.

Pupils should spend time playing their game to ensure that everything works as expected and matches their original plan. They should try to debug their game where necessary, pupils can support each other in this task.

The class will peer asses their games using two stars and a wish in a similar way as they did with their story. Pupils should swap computer with a partner and play their partner's game. Pupils should then identifying two things they liked or thought had been done well in their partner's game and one things that they think could be improved upon.

At the end of the lesson, pupils should pass their peer evaluation to their back to their partner to allow them time to further debug their game and make any suggested improvements.

Plenary

Pupils will work with a partner to evaluate each other's games and stories. They will test them to find bugs and help each other to debug if necessary. They will check that the final game meets all objectives.

At the end of each lesson discuss learning with pupils, ask questions about topic of lesson.

Pupils can work with a partner to share what they have learned.

Differentiation

Pupils can make use of teacher examples of code to copy if additional support required. Some pupils may require additional support.

Pupils can extend game to include additional functionality including further use of subprograms and adding more game levels.

The level of support provided to pupils will be differentiated.

The timing for each lesson can be flexible depending on how pupils are coping with the activity.

Assessment Opportunities

- Pupil has used constructs in Scratch to meet benchmarks (use of subprogram, multiple variables, complex conditions).
- Scratch game runs and pupil has successfully identified and fixed bugs either independently or with support of a partner.

Pupils will be observed creating games and supported through the process. Progress can be assessed through observation.

Pupils will peer assess games and story writing to check that they meet the required criteria.

Pupils can explain purpose of code to teacher and to peers.

Pupil's learning can be assessed by documenting code using screenshots and having pupils explain the purpose of a block of code.