Year 2 Habitat scene - Lesson 1

Duration: 60 mins

Overview

In this series of lessons pupils create a simple interactive scene in Scratch. In the first lesson, pupils design their scene and explore example code which will be helpful to them in coding their work. In the second lesson they program and debug their code. In the final lesson they evaluate their work, making improvements and reflecting on what they have learnt.

Objectives

Computing

- I can design a habitat scene and label it.
- I can explore code to find useful commands.
- I can add action to my scene.
- I know my notes about the action are algorithms.

Science

- I can describe some habitats and the plants and animals that live there.

Before you start and what you need:

- Adapt the presentation to your school's format as required.
- Print the design sheets.
- Review the example designs to give you ideas for modeling how to create a design.
- Provide pupils with access to the exploration example file.
- Provide each pupil with a design notebook, for jotting down useful commands in.
- If required, find animations or websites to support pupils revision of habitats e.g.
  http://www.bbc.co.uk/bitesize/ks1/science/plants_and_animals/play/
  https://www.tes.co.uk/teaching-resource/habitats-presentation-for-the-children-6138409
- Perhaps look at this example scene before the lesson, to give you some ideas
  http://scratch.mit.edu/studios/1066000/

Lesson Introduction - design (10 mins)

- Ask pupils what animals need to live and what we call the place they live (**habitat**).
- Explain to pupils that over the next three lessons they will be creating a scene, in Scratch, to show what they know about a habitat.
- Explain that a simple way to design a scene is to draw a diagram with labels.
- Choose a particular habitat for your scene, in this example the seashore has been chosen. Using input from the class for ideas, model how to draw the habitat, adding a few animals, food sources, shelter etc. (Slide 3/4). Keep the scene simple, using simple drawings. At this stage do not introduce movement, sound or text. Pupils will add this detail later.

Pupils Task - Design (10 minutes)

- Ask pupils, in pairs, to create their scene designs. Both pupils may create their own version of a common design, or they may create a single shared design.
- Remind pupils to create a sketch with labels rather than detailed artwork. **Note** - as pupils come to write their code their designs will change and develop. If their initial designs are very detailed they may be reticent to develop and adapt their work.

- Ask pairs to share their design with another pair. Encourage pupils to evaluate their peers' work, perhaps building a class list of what makes a good scene (simple clear sketches, labels to show the things in the scene). Give pupils time to improve their work.

Exploring examples for useful commands (10 mins)

- Once most pupils have completed their design, stop the class and ask them how we might create an scene. *Use ScratchJnr, Scratch (or other programming languages) running on a computer.*
- Ask pupils to think-pair-share what they can remember about using the software from any previous programming they have completed.
- Explain that they will now explore an example program to find commands which might be useful for creating their scene. Ask pupils to use their **Design Notebook** to note down any useful commands or blocks of commands they find in the example code that they might be able to reuse in their animations.
- Model how to open the example file and make notes, ask pupils to do the same and explore the scenes.

Review of exploration (10 minutes)

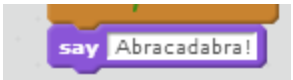
- Ask pairs of pupils to compare what they have recorded in their notebook and discuss the useful commands they have found.
- Ask one or two groups to share what they have found, showing the example code to the class using the interactive whiteboard. Discuss as a class how these might be used in their scenes.
- Ensure that the following commands have been recognised by pupils and included in their notebook.
    - The '**when**  **clicked**' command runs the script (set of commands) when the green flag is clicked.

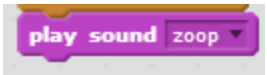
    - The **'move'** command makes a sprite move. The **'wait**' and **'point in direction**' commands are also useful here.

    - The **'say '** command displays text in a speech bubbles. This can be used to give a useful fact.

    - The '**play sound**' command adds a sound.

Updating the designs - adding action - algorithms (10 minutes)

- Model how to add action to your design (slide 6).
- Give pupils 10 minutes to add some simple action to their designs, e.g. the crab moves across the scene, a fish makes a bubble sound, a seagull says 'Yum fish!'.
- Explain to pupils that the actions they have added are a set of steps to make something happen. Can they remember what we call this? (**Algorithm**).

Plenary (10 mins)

- Ask two or three pupils/groups to present and talk through their designs. Encourage other pupils to question the groups on their designs and comment on them - what do they like? Why? What do they think they could make even better? Why?
- Review the learning outcomes for this lessons from slide 2 of the presentation.

Differentiation

Ideas to challenge pupils:

- Add extra annotation or notes to the design to improve precision and accuracy e.g. number the movements and action on the scene. Make a sequence of events, where events occur one after the other. For example, 1. Seagull says 'Yum fish'', 2. Crab waits and then moves. 3. Fish waits and makes a popping sound after crab has finished moving. Perhaps use 'wait' commands to synchronise events.
- Add extra complexity to design such as a repeat e.g 'crab moves back and forth 3 times and then disappears'.

Ideas to support pupils:

- Reduce the number of animals or plants in the scene to just 1 or 2.
- With adult support create a group scene design.
- Provide pupils with partially completed scene, that pupils complete (habitat starter).

Assessment

Emerging	**Expected**	**Exceeding**
Requires support to work out what is most important to include in their design (labelled scene). Knows an algorithm is a set of steps that makes something happen. Requires support to try out example code or find useful commands.	Creates a design (labelled scene), labels the background, objects and adds action. Knows the notes about action are algorithms. Runs example code and pinpoints one or two useful commands.	Adds extra annotation to their design (labelled scene) to improve precision. Adds extra complexity to the action, such as a sequence of events. Confidently reads code and traces it to the actions it performs and pinpoints most useful commands.

Adapting the planning

Create scenes for other situations or things , e.g. story scenes, places in geography, diagrams of objects in science etc.