



# My First Independent Scratch Project



3 hours  
Approximately 40 minutes design and the rest for programming

**Aims**  
Pupils design and make their first simple Scratch project.

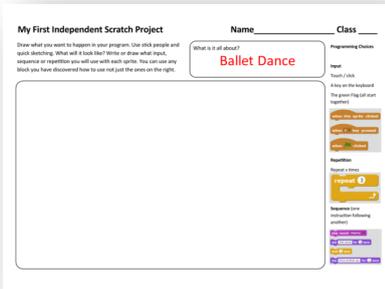
**Computing Concepts**  
-Algorithmic design (what will my sprites do and what order will they do it in?)  
-Appearance Design (What will my program, sprites and background look like?)  
-Input -sequence -repetition

Explain to the class that after learning how to create inputs, sequences and use repetition through making (insert names of projects) they are now going to design and create their own project.

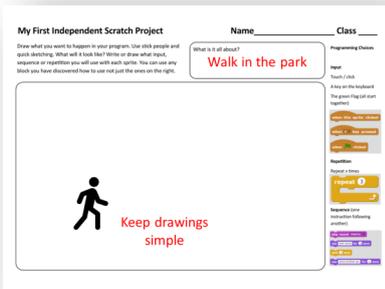


Open up the slides **how to plan my first independent Scratch project** shown on the left.

Show pupils the planner slide 2. Read the instructions on the top.



Slides 3-6 have simple example project names. Make sure you say that they will be able to think of better projects.



Slide 7 is a reminder to keep their drawings simple.

Slide 8 is a demonstration of how to plan what their program will do. What sort of input will they use? What will that input start? How long will it last for?

**Question** What is missing in this planning?

**Answer** Timings on the SAY command

The screenshot shows a planning sheet titled 'My First Independent Scratch Project' with a 'Name' and 'Class' field. The project title is 'Walk in the park'. The 'What is it all about?' section contains a drawing of a person and a dog, with instructions: 'Click Say Nice day for walk in the park', 'Up Arrow Move', 'Right Arrow Turn Right', 'Left Arrow Turn Left'. The 'Programming Choices' sidebar on the right shows 'Say' selected.

Slide 9 is a reminder to plan what will go into the background. Remember it is only a plan so don't spend too much time drawing it.

This screenshot is similar to Slide 8 but includes a 'Draw or List what the background will look like' section with the following items: 'Grass', 'Trees', and 'Playing Sounds'. The 'Say' command in the programming choices is now greyed out.

Slide 10 is a final reminder to keep it simple.

This screenshot is similar to Slide 9 but the 'Draw or List' section is simplified to 'Keep it simple.'. The 'Say' command in the programming choices is now greyed out.

Give pupils 20 minutes of dedicated design time. Those who need longer can have more time but others can start turning their design into a program.

Once pupils are programming tell them all that they can only have one teacher code hint so they need to make sure that they use that sensibly.

If they have a problem that is not code related such as *my copy of Scratch has frozen* they should say straight away.

Also instruct them not to help each other as this is their own work.

They may use the sequence card and the repeat x times loop card (Print 5 copies of each, fold them up and leave them on the side.) and refer to their sequence and repetition sheets, if you have given these out previously. They may also save their work and look at any of their previous Scratch projects.

At some point say that it is ok if their program looks or acts differently from their design as everyone adapts their design while programming.

As they are planning go round and make sure each idea is simple enough. There is often one or two pupils who want to recreate Minecraft without realising that this took many programmers years to create.

*I recognise there is more than one way to solve/describe a problem*

*I can evaluate my solutions against a set criteria*

*I can design criteria to evaluate my creations*

*I can contribute useful ideas to a partner or group*

*I can encourage others to share their ideas*

*I lead using all the people talent in my group*

*I learn from setbacks and don't let them put me off*

*I can persevere even if the solution is not obvious*

*I don't just accept the first solution*

*I look for a range of solutions to the same problem*

*I look for how a project can be extended*

*I can break complex problems into parts*

*I can discover/concentrate on the most important part of a problem*

*I can identify patterns in problems and solutions*

*I can adapt existing ideas to solve new problems*

*I can develop, test and debug until a product is refined*

*I make predictions about what will happen*

*I repeatedly experiment through predicting, making, testing & debugging*



Adapted from a problem solving rubric created by Mark Dorling & Thomas Stephens hosted at <http://code-it.co.uk/attitudes/>

At the end of the project you could ask pupils to circle all the computing attitudes they have used.