Level 6

Frantic Felix

CODE CLUB

NOTES FOR CLUB LEADERS

Introduction:
In this project the students will make a platform game like Manic Miner or Mario Bros. This project will be the hardest thing they have tackled so far and it won't be a walk in the park for them so brace yourself! This project should take 3-4 weeks.

Resources
This project uses assets provided by Code Club so check your folder. Children will create other assets themsleves.

Scratch Cards required:
Record a sequence
Play back a sequence

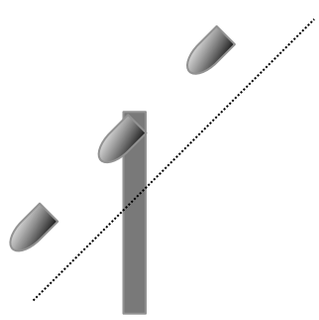
Notes on triclky bits and collision detection:
Platform games, like Manic Miner and Mario Bros, are all about the central figure moving around a world and bumping into things. Some things, like walls and floors, just stop you moving. Some things, like baddies, will kill you. Some things, like keys, you want to collect. Some things are just scenery and don't affect the game at all.

That means that knowing when you've bumped into something (called 'collision detection' - you might need to explain this to your students) is important. Scratch gives you a few blocks for collision detection: <touching? [sprite]>, <touching color []?>, <color [] is touching []?>, and (distance to [sprite]). However, to make the game, you need to know more than just whether you're touching something: you need to know which side is touching it. If you walk up to a wall on the left, you can't go left any more.

But you can still go right, or jump, or fall if there's no floor. Touching a baddie may hurt you; but if you touch it with your feet, you may hurt the baddie. None of the built-in Scratch blocks tell you about direction of contact. So we've got to build our own collision detectors.

Collision detectors: The bullet collision problem

You may have noticed that Felix moves quite slowly. This is to prevent something called the bullet collision problem. If you have a game that involves shooting bullets into thin targets, what can happen is the the bullet moves through the target without hitting it. This is because the computer moves the bullets in steps and checks for collisions after each step. If one step is before the target and the next after it, the computer will never realise that the bullet has hit something (see the diagram below).



The collision detectors used in this game suffer from the same problem, as they're quite thin. If you increase the distances Felix moves at each step (try 20 or more) and then drag Felix into different places, you'll find that he can sometimes drop through the floors!

You can stop Felix moving through thin things by making the collision detectors larger, but that doesn't stop the problem of Felix moving into the middle of things before Scratch realises that he's bumped into something. There are ways of getting round it (such as keeping everything to a grid, or moving Felix back a bit if he's bumped into something, or predicting where he'll be and moving slowly up to it) but they're either complicated or slow or both. I'll stick to a fairly leisurely Felix in this project. If you want to explore other alternatives, go for it!

Drawing your own collision detectors

If the children want to use their own costume instead of Felix, they'll have to create their own collision detectors for it. I created them by importing the costume into the GIMP and creating one layer for each side bar. By selecting which layers are visible, you can easily export all four collision detectors (a .gif files). As when designing levels (below), check that you get the correct colours when you convert the image to an indexed palette.

Also note that the top and bottom edge detectors shouldn't overlap the side ones. This prevents the sprite hanging in mid-air if it bumps into an obstacle during a jump.

Task 4: Keys and goals

You may wonder why we're putting in explicit starting positions for the keys, the pod, and Felix, and why we're sending messages on a win. It's because when you make your own levels, you'll want to have these things appear in different places in different levels. If we put the hooks in now, it'll be easier later to make the levels appear correctly.

Week 2: Designing levels

If you use a drawing package for creating levels (rather than Scratch's built-in one), beware of when you convert it to a GIF. Moving from RGB-based images to palette-based images can make some of the colours change slightly, though not enough to see with the naked eye. That means that some of the Scratch colour-detecting blocks will no longer work. You may need to manually edit the colour palettes to preserve the colours you want.

Make sure the children note all the information they'll need to set up each level. They'll need:

Felix's starting x and y values

The pod's x and y values

The x and y values for each key

(list from previous page continued)
Whether a key is visible in a level
How many keys are in a level
The background image for the level
For each baddie, they need
The starting x and y values
The final x and y values
The initial direction
Whether it's visible
Whether it moves horizontally or vertically
It's costume

Week 3

Task 1: Displaying a new level

Be aware, this is a long task. There are lots of changes to make. Just about every sprite needs to respond to the start level message. Every sprite will need several lists, specifying its initial values in each level.

Encourage the children to do one sprite at a time, testing as they go. We have used a couple of script fragments on the Stage to set the current level then broadcast start level. If they're doing multiple keys or baddies, encourage them to delete all but on of each type of sprite, make the modifications, then duplicate the sprite and change its parameters.