Make your own Temple Run game...

These instructions will talk you through how to make your own Temple Run game with your pupils. The game is made in Scratch, which can be downloaded here: [http://scratch.mit.edu](http://scratch.mit.edu). You will also need to have downloaded the image pack containing all the graphics required to make the game. This is available from [www.primarycomputing.co.uk](http://www.primarycomputing.co.uk).

In making the game, pupils will learn about control inputs, variables and conditional statements. The screenshot below is off the completed game with the Temple Run character jumping a hole! The aim of the game is for the runner to run for as long as possible without falling down a hole and turning into a ghost!

First let’s get everything into Scratch that we’ll need to make the Temple Run game.

**STEP 1:** Open Scratch and remove the default sprite Felix. Choose ‘Select new sprite from file’ and choose the ‘Sprite1Run’ image from the image pack. Edit the costume by using the rubber to remove the bricks from where I have cropped the image from a screenshot on
my iPad (see image). Once you have done this, we also need to import two costumes for the runner. This can be done by selecting the costume tab and clicking import. The first costume to add it called Sprite1jump and secondly add a ghost costume from Scratch’s library.

**STEP 2:** Next we’re going to add the second sprite, which in this game is actually the scrolling background. Select add sprite from folder and add Scrolling Background. Once the scrolling background is also in Scratch it may sit over our runner sprite. If this happens at any point you can right click on the runner sprite in the box bottom right of the screen and select ‘show’.

**Step 3:** The final graphic we will need is the background. This might seem a bit odd at first since we have loaded our scrolling background above, but it will become apparent why we need an additional background shortly. To load the background, select the stage icon, click backgrounds and import the image titled background.

We now have everything we need to start programming our game.

**Step 4:** Let’s program the runner first and start by making him able to jump the holes in the path ahead. Position your runner in the middle at the bottom of the screen as shown here. To make him appear to jump (each time the spacebar is pressed), we are going to get him to move forward (vertically) briefly and change costume to a jumping pose. Since gravity will quickly get the better of him, we want him then to return to his original position and switch back to running mode. The coding blocks to achieve this are shown below.

Once you’ve added this code, try pressing spacebar and your runner should look like he jumps! Try altering the time and distance values. How does this affect his jump?
Step 5: Now let's program the background. The background is going to move in two directions: it will move vertically downwards in the y direction to give the illusion that our runner is running forwards and it will move in the x direction (left and right) to give the illusion that our runner is moving left or right. Let us consider the vertical movement first.

Vertical movement: To start this game a key will be pressed which will start the runner running along the path. In actual fact what will happen is that when the key is pressed, the background will start scrolling down and give the illusion that the runner is running forward. To achieve this you need to create a variable called 'scroll' for all sprites. Then within the script area for the runner add the blocks below:

```
when a key pressed
forever
change Scroll by -5
```

This will mean that once the ‘a’ key is pushed, the variable ‘Scroll’ will continually decrease in -5 steps.

If we now go back to the script area for the scrolling background we can use this variable to control the background position as shown using the blocks below. Here the y value of the scrolling background is being set to the variable scroll, which we have just set to steadily decrease once ‘a’ is pressed.

Please ignore the + 360*0 for the moment we will come onto this.

The blocks on the left are used to reset things when the green flag is clicked. Here you can see the scrolling background is set to x=0 and y=0.

```
when clicked
go to x: 0 y: 0
set Scroll to 0
```

```
when a key pressed
forever
set y to Scroll + 360 * 0
```

Once you have completed this step, you should have a scrolling background although the background scrolls out of screen pretty quickly! We shall fix that shortly by duplicating many more backgrounds to scroll through in turn.
**Horizontal movement:** As well as scroll vertically, we need the background to scroll left to right (in the x axis) to give the illusion that our runner can move left or right. To make the game more accurate we also need this movement limited to within the walls of the path. The way I tackled this problem was to make a variable called x and then added the following code to the runner’s script area:

![Code for horizontal movement](image1.png)

And this code to the scrolling background script area

![Code for scrolling background](image2.png)

If you read through the code, you will see that movement left or right is limited when the x variable exceeds certain values (35<x>35). This causes the runner to stay within the boundaries of the wall. The value of 35 was worked out through trial and error! It should also now become apparent why we needed that second background as when our scrolling background moves it would just leave white at the end of our screen if we didn’t have the second background behind.

**Step 6:** The next step is to extend the scrolling background so our game lasts longer and also add some dangerous holes for the runner to fall down. To do this we are going to use the copy sprite function - this is accessed by right clicking on the scrolling background and selecting ‘duplicate’. The more times you duplicate your scrolling background, the longer your game will be! A good starting point is 10 duplications.

*We now need to make some changes to each scrolling background.*
Firstly, on each one, use the paint editor to add some dangerous holes by using the black circle tool, like the hole that’s shown on the path at the very start of this document. Don’t make the holes too big – about the size shown. To make your game more tricky, make the holes in different places on all of the duplicates you have just made of the scrolling background.

You now need to change a very small amount of the code on each scrolling background to help align them all so they scroll in aligned and one after the other. To achieve this, on each of the duplicate slides change the value of the number shown in the circled box below to increase by 1 each time. So, as we have shown above, the first value was 0, on the second scrolling background it will be 1, then 2, then 3 etc.

At the same time you are making the change above, change each of the y values in the code shown below on each background to increase by 360 each time. This is because the height of the program area is 360 pixels, so by increasing the value by 360 each time then when the green flag is pushed, all the backgrounds line up one after another ‘behind the scenes’.

Step 7: We are now going to write the code so that the runner is able to jump the holes. We have already written the program for him to be able to jump, so here we shall tell Scratch that ‘if’ he is jumping when he touches the colour black (which is while we made the holes from black circles) then nothing should happen but ‘if’ he is not jumping then he should turn into a ghost and the game stops since he has fallen into the hole. The code for this is shown here to the left.
You will notice that a score variable has also been incorporated.

In this game, players scores increase for the longer they can run for. This was achieved with this simple coding block below.

![Coding block](image)

You should now have coded the Temple Run game! I hope you have fun playing it 😊

Why don’t you try this extension?

**Extension**: Could you develop the game so the runner collects gold coins to increase their score like in Temple Run on the iPad?

**Hint**: If the coins were a specific colour you could use the ‘when touching colour’ block...