1.4 PLATFORM GAME IN SCRATCH

This lesson will teach you how to create a simple platform game in Scratch:

What you will learn…

- How to create **gravity** in your platform games
- How to use the **broadcast** blocks to coordinate or synchronise events between sprites
- How to create **random numbers** – these can be very useful for many coding projects
- How to use **event handling** to control your games
- How to make your game characters **jump**

You can have a look at our project *Evil Daddy* now. You'll find it on the Nerdsville Scratch page or follow this link. Have a go – try to sneak the boy past Evil Daddy and get to the TV remote control before Daddy catches him and makes him do his homework.

http://scratch.mit.edu/projects/26107515/

After playing the game *look inside* to see the code. You'll see that Evil Daddy has one backdrop and four sprites.

When you are ready to create your own version *Sign in* to Scratch and click on *Create*. Make sure you give your new project a name.

STEP-BY-STEP INSTRUCTIONS – CREATE YOUR OWN PLATFORM GAME

BACKDROP

The backdrop for Evil Daddy is based on photographs of rooms in my house – you can create your own backdrop based on an image from the Scratch library, your own painted backdrop or an uploaded image. Notice the fluorescent green borders, floors and stairs – these were added using the Scratch paint editor. The scripts in this project rely on the detection of collisions between the sprites and this green colour.

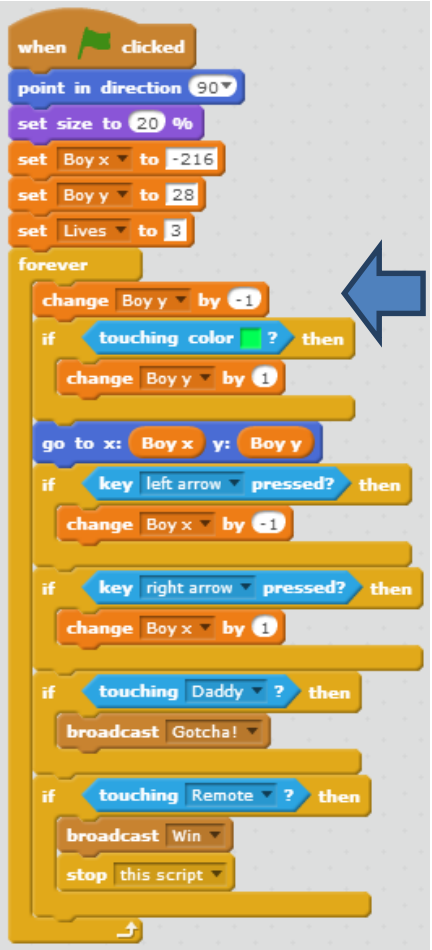
- Choose, paint or upload your basic background image ☐
- ✓ Go to the backdrop paint editor and paint your 'platforms' or 'floors' the colour of your choice

The *Evil Daddy* backdrop –The borders, floors and stairs are all painted the same fluorescent green colour. This makes it easy to create code that detects when our sprites are touching these objects.

THE BOY (MAIN CHARACTER) SPRITE

The Boy sprite is from the library of Scratch 1.4 (an older version of Scratch).

- Create your main character sprite – paint it, choose from the library or upload a file ☐
- Now create your character's main script. Here's the Boy sprite main script… ☐



Look carefully at the script. Make sure you know what each block does.

During the initial setup we create three variables: *Boy x*, *Boy y* and *Lives*. These variables are used to set Boy's x and y coordinates for the start of the game and keep count of how many lives he has left. Lives is initially set to 3.

This section simulates **gravity** – every time the forever loop repeats, *Boy y* is changed by -1 – making him 'fall' down the screen. However if he's touching the green floor, *Boy y* is increased by 1 so he doesn't fall through floors.

Next, a `go to x:Boy x y:Boy y` block moves the boy to his set coordinates.

Two *if* statements are used to detect key presses and change the *Boy x* variables accordingly.

Finally, two *if* statements are used to detect collisions between the Boy sprite and the *Daddy* sprite or the *Remote* sprite. If one of these events occur, a `broadcast_` block is used to send a message to all sprites in the project. You create your own message in the drop-down box. You'll learn more about how the broadcast blocks work below.

(We could have used the `change x by_` and `change y by_` blocks instead of creating variables for Boy's coordinates, however it's good for you to know that there are often different ways of achieving the same result. As long as your code works and isn't buggy then any method is fine!)

- Create a script to handle what happens to the main character sprite when *Gotcha!* is broadcast… ☐

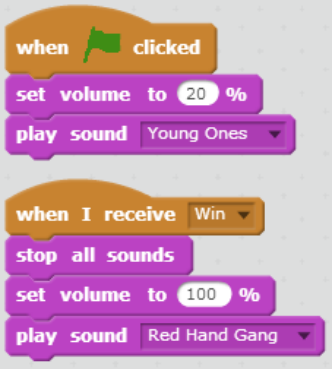
In Boy's main script the *Gotcha!* message is **broadcast** when he touches the *Daddy* sprite. This small script makes the boy spin around when the message is received.

The `when I receive_` block is used to create a new script that will run when a particular message is received. The drop-down box allows you to choose the message. In this script the attached blocks cause the boy to spin, then point in direction 90 again.

There is a second `when I receive Gotcha!` script for the *Daddy* sprite that plays the "Do your homework!" sound and handles what happens to the *Lives* variable and what happens when Lives = 0.

You can create a number of scripts for different sprites that will all run at the same time when a message is broadcast. The `broadcast_ and wait` block can be used to trigger a number of other scripts, then wait until they have finished before continuing.

- Create the main character's sound scripts… ☐



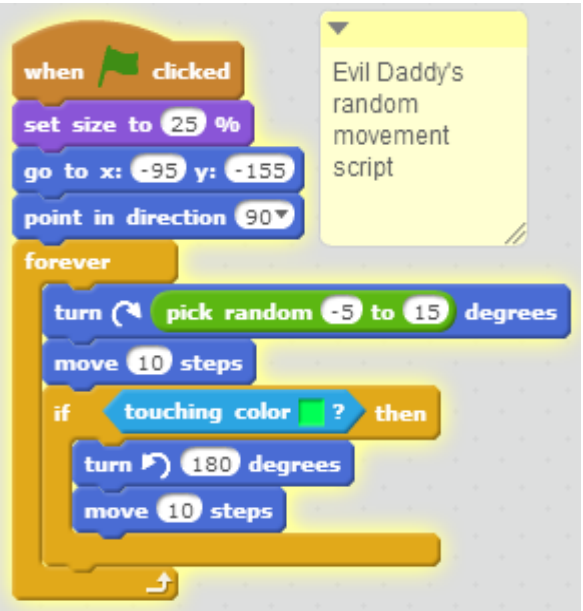
I've used two of my favourite TV themes for the soundtrack – the first, from the Young Ones is the main theme and plays when the green flag is clicked.

The Red Hang Gang theme plays when the boy succeeds in turning on the TV – along with the show's image sprite appearing on the TV screen. Both of these events are triggered by the *Win* message being broadcast.

THE DADDY SPRITE

The *Daddy* sprite is from the library of Scratch 1.4 (an older version of Scratch).

- Create your 'baddie' sprite – paint it, choose from the library or upload a file ☐
- Now create the sprite's main script. Here's the Daddy sprite script… ☐



After setting the size, start position and direction of the sprite a forever loop is set up that contains Daddy's movement script…

Daddy is set to turn randomly and then move 10 steps each time the loop repeats. The **random number** block from the *Operators* palette is very useful for creating random sprite behaviour in games. Experiment with the range of numbers until your sprite moves in the way you like.

An *if* statement is used to check if Daddy collides with a green wall or floor and make him bounce off (turn 180 degrees and move 10 steps) if he does.

Daddy's second script is triggered by the broadcast of the *Gotcha!* message when the Boy sprite touches the Daddy sprite (see the Boy sprite main script above).

- Create a script handles what happens to the 'baddie' sprite when *Gotcha!* is broadcast… ☐

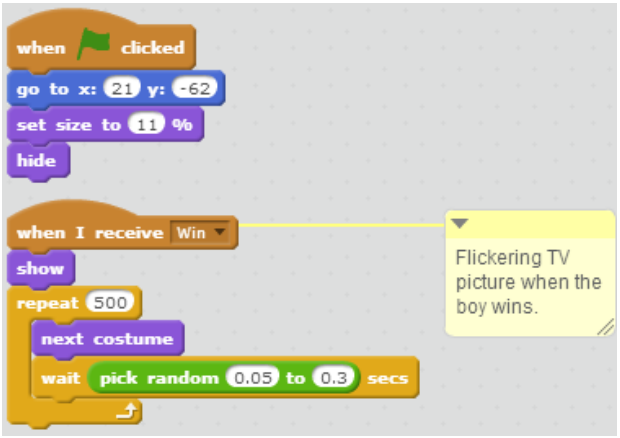
On receiving the *Gotcha!* broadcast, this script plays the *Homework* sound ('Do your homework!' – recorded by me doing a funny voice close to the computer's microphone) and changes the *Lives* variable by -1.

Then an *if* statement checks if *Lives* is less than 1 (this is better than using 'if Lives = 0' just in case two lives are lost quickly). If Lives *is* less than 1 then Lives is set to zero (so you don't end up with the silly situation of having a negative amount of lives), all sounds are stopped, volume is set to 30% and the *wahwah* sound (me again!) is played. After 3 seconds all scripts are stopped – the game is over.

THE TV SCREEN SPRITE – 'GAME OVER'

The TV screen sprite has two small scripts. When you design your own game you can choose to end it however you like – perhaps choose your own sound or tune and create a 'Game Over' message sprite or make your sprites spin around or disappear into the distance.

Create scripts that handle what happens when it's 'Game Over'… ☐



The first small script sets the position of my *TV screen* sprite and then *hides* it when the game starts.

The second script is triggered when the *Win* broadcast is received. The sprite is shown and then made to randomly flicker between two costumes. The two costumes were made slightly different to give the illusion of a flickering TV screen.

THE REMOTE SPRITE

The remote control sprite simply sits there waiting to be touched by the Boy sprite in order to win the game (the code for this is in Boy's main script).

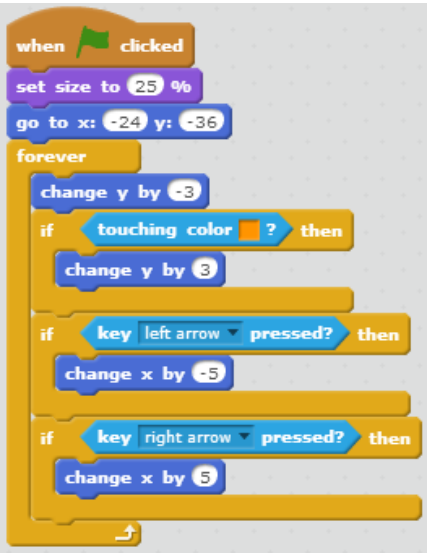
Here's the code.

STEP-BY-STEP INSTRUCTIONS – A PLATFORM GAME WITH JUMPING

Have a look at our *Cubit Platformer* project here…

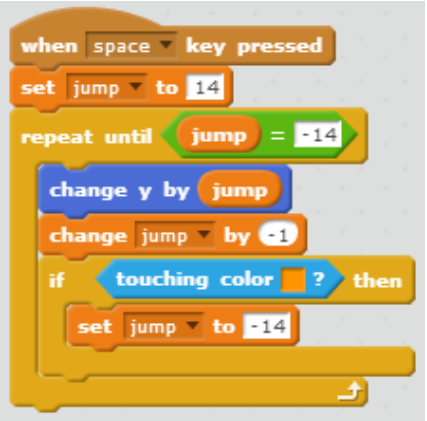
http://scratch.mit.edu/projects/23593836/

It's not a complete game, but simply a basic template for a platform game with jumping. We've used the Nerdsville logo, *Cubit* as our only sprite and drawn a background with a floor and platforms of the same orange colour.



This is the main script – it should all look quite familiar and is similar to Boy's main script in Evil Daddy.

After setting size and initial position, a forever loop sets up gravity and monitors for the left and right arrows being pressed to change the sprite's horizontal position.



This second script uses **event handling** to wait for the space bar to be pressed and trigger a script that makes Cubit jump. *Event handling* is commonly used in computer languages where code needs to respond to a certain event – like a key press or a mouse click.

When *space* is pressed the variable *jump* is given the value 14, then a loop is set up in which the sprite's y value is changed by *jump*, and jump's value decreased by 1. This has the effect of making Cubit jump, decelerate and then accelerate downwards again.

A `touching colour_?` block is used to end the jump if the sprite is touching the orange ground / platform colour.

You should now be able to experiment with creating your own platform game with a jumping main character sprite and 'baddies' that move randomly. Have fun!

What you have learned…

- You should now be quite good at reading somebody else's code and figuring out how it works – it's a handy skill and you've had a lot of practice
- How to include **gravity** in your games and make a character **jump**
- The **broadcast** blocks are very useful for coordinating and synchronising sprites
- **Event handling** allows you to write scripts that run in response to particular events

NOW TRY THIS…

Once again, feel free to remix the Nerdsville projects.

- Now might be a good time to explore some of the six-million plus projects that other Scratch users have shared – you can *look inside* and see how their projects work - this is a great way to learn.
- Make sure you ask your teacher for permission if you'd like to do this in class.
- Playing Scratch games might be fun, but make sure you also figure out how the code works!
- You can remix projects or 'borrow' scripts and graphics from them using your backpack.
- Make sure you include credits on the project page if you use other people's material.

QUIZ

Question 1 – Why is it handy to have platforms the same colour?

a) They look prettier that way
b) So that the `touching colour_?` block can be used to detect when a sprite is touching them
c) Different coloured platforms are not possible
d) Because platforms are all made from the same sprite

Question 2 – How do we create the illusion of gravity?

a) Change the y position of a sprite by a positive amount each repeat
b) Change the y position of a sprite by a negative amount each repeat
c) Change the x position of a sprite by a positive amount each repeat
d) Change the x position of a sprite by a negative amount each repeat

Question 3 –When might we use a *broadcast* block?

a) When we are making a TV sprite
b) When we want to send a message to another sprite's script

c) When we want to detect a key press
d) When we want to detect a mouse press

Question 4 – Random numbers are handy for…

a) Making a game character move in a crazy, random way
b) Creating number guessing games
c) Making sprites appear in unpredictable places on the stage
d) All of the above

Question 5 – Which of the following can use event handling?

a) Detecting when two sprites touch
b) Making a 'Game Over' message appear
c) Making a key press cause a character to jump
d) Sending a message to other sprites in a project

Log in to **www.giantclassroom.com.au**to enter your answers.