Week 4

Last week we made it so the player can put ‘direction instruction’ arrows onto the reactor grid, and move the arrows around. Next we’ll add a ‘robot’ to follow those instructions.

Either remix my ‘end of Week 3’ project, or start from where you left off last week.

Add the ‘Waldo’ sprite

In SpaceChem, the two robots are called Waldos. Choose a sprite from the library and rename the sprite to ‘Waldo’. Depending which sprite you choose, you might need to adjust the ‘rotation style’. I used ‘Ladybug 1’.

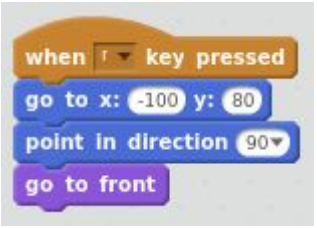
Shrink it until it fits properly inside one of your reactor cells.

Start Waldo off correctly

We’ll need a way to put Waldo into the starting cell. By checking what numbers you used in the cell’s ‘go to x (–) y (–)’ block, you should be able to work out the x and y numbers for your chosen starting cell. Or you can carefully put it right in the middle of a cell and drag out the ‘go to x (–) y (–)’ block Scratch fills in for you.

Use ‘when key pressed’ for testing

One good trick when working on something new is to make it happen ‘when key pressed’, so you can test it more easily. So let’s make a ‘reset Waldo’ script. ***Add* this script to Waldo:**



It puts Waldo back to the start, facing rightwards, and in front of the reactor cells. I used the ‘r’ key, for ‘**R**eset’. You’ll probably need different numbers instead of my ‘−100’ and ‘80’.

Make a ‘step one cell’ block

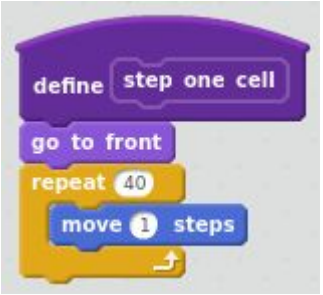
As we learnt last week, we’ll create our own block which makes Waldo move one ‘step’, from one cell to the next. So click the Make a Block button in the More Blocks section, and call your new block something sensible — I called it ‘step one cell’.

Our reactor is a regular grid, so moving Waldo from one cell to the next is fairly simple: Just move however many Scratch units apart your cells are. This is the number you multiply by when working out where the cells go. My ‘cell’ sprite has

*go to x ((**40** * cell-x) − 100) y ((**40** * cell-y) − 120)*

so my cells are **40** apart.

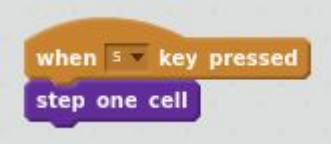
To make it look better, we'll make Waldo move in lots of short moves, not one big one. So define your 'step one cell' block



***Add* blocks to the 'define' hat-block to get this script**:

And we'll use the same trick of using a keypress to let us test our scripts.

***Add* a script to use our new block when 's' is pressed**:



Try it: Press 's' (for '**S**tep') — Waldo should move one cell.

Check you can still reset Waldo by pressing 'r'.

Challenge

Can you make Waldo move more quickly? Hint: It needs to move 40 steps altogether, but it doesn't have to be in 40 moves of 1 step each.

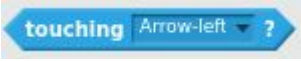
Make Waldo obey instruction arrows

But if you go full-screen and put some direction instruction arrows onto the grid, Waldo just ignores them.

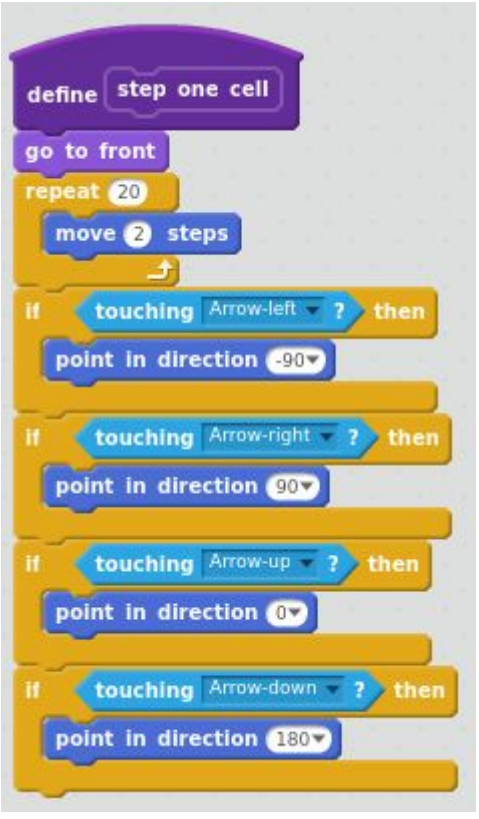
After Waldo has moved to a new cell, it needs to:

- Check whether the cell has a direction instruction arrow on it.
- If so, point itself in the instructed direction.

To tell whether there's an arrow, we can use blocks like



, so we just need to check each arrow in turn, and turn round if we're touching (a clone of) that arrow.



***Change* your 'step one cell' definition to check the four arrows**:

I gave my arrow sprite sensible names, so this code is easy to read. If yours are still 'Arrow1', 'Arrow2', etc., it would be worth changing their names first.

You can use the drop-down arrowhead in 'point in direction' to choose the numbers.

Try it!

Go full-screen, drag some arrows onto the grid, and keep pressing 's' to check Waldo follows them.

Taking lots of steps

Now this is working, there's not much more to do to make Waldo march around by itself, following the direction arrows. We just need to keep doing 'step one cell', over and over.

***Add* a script which keeps taking steps forever**:

I made this happen when you press 'g' for '**G**o'.

Try it!

In full-screen, drag some arrows onto the reactor grid which will make Waldo go round in a loop, and check that it works when you press 'g'.

Stop Waldo when you 'reset' it

Set Waldo going by pressing the 'g' key, and then try to reset it by pressing 'r'. Waldo will go back to the start, but then immediately start moving again. The problem is that the 'forever: step-one-cell' script keeps going. We can fix this by making the 'reset' script stop everything else.

***Change* the 'when 'r' key pressed' script by adding a 'stop' block**:

You need to choose 'other scripts in sprite' before you can connect the 'stop' block into your script.

Now Waldo should properly reset, even if it is in the middle of moving when you press 'r'.

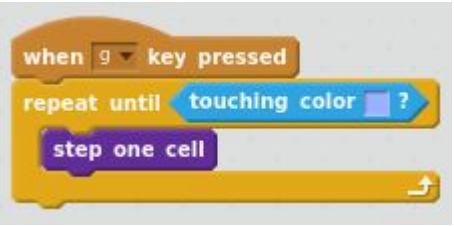
Stop Waldo if it goes out of the reactor

If you don't put some arrows in to turn it round, Waldo will just carry on marching right out of the reactor. This is no good.

The easiest way to tell when Waldo has gone outside the grid is to make the stage a particular colour, and check for that colour.

***Change* your stage's backdrop** to be one big rectangle of a colour *which isn't part of your arrow*. I used a pale blue.

Now, instead of stepping 'forever', we keep stepping only until we go outside the grid.



***Change* your 'when 'g' key pressed' script** by replacing 'forever' with 'repeat until':

For the 'color', click on the little square of colour, then with the 'pointing hand' cursor click on your stage's new colour background.

Check that Waldo now stops if it goes outside the grid.

Add buttons for ‘go’ and ‘reset’

Now things are working, we’ll add buttons for the player to click, instead of having to press the keys ‘r’ or ‘g’.

***Add* one sprite for the ‘stop’ button, and another one for the ‘go’ button:**

I used ‘Button3’ from the library, changed the colour, and added text, but you can choose your own. Adjust the size and put them somewhere on your game stage.

Sending messages with ‘broadcast’

In Scratch, to get one sprite to tell another sprite what to do, a very good way is the ‘broadcast’ and ‘when I receive’ blocks. When the ‘stop’ sprite is clicked, it will broadcast a message, and the Waldo sprite will listen for that message.

***Add* this script to the ‘stop’ button sprite**:

To make the ‘reset Waldo’ message, you have to click the drop-down arrowhead of the ‘broadcast’ block, choose ‘new message...’, and then type in ‘reset Waldo’.

Now instead of making Waldo do its reset work when we press ‘r’, we need to make this work happen when Waldo hears this message being broadcast.

***Change* the hat-block of Waldo’s ‘reset’ script**:

You should now be able to reset Waldo by clicking the button.

Challenge: Make the ‘go’ button work

Try to make the ‘go’ button work in the same way, so the player can click ‘go’ instead of pressing the ‘g’ key.

Hint: Make a ‘start Waldo’ message, and broadcast and receive it.

Add ‘fast’ and ‘slow’ speed buttons

If you did the challenge about making Waldo move more quickly, you’ll have worked out that we can do ‘repeat 20: move 2 steps’, or ‘repeat 10: move 4 steps’, or a few other combinations which make 40 steps altogether. We’ll add buttons to let the player choose how fast Waldo moves.

Remember how big one ‘stride’ is

We now need a variable to keep track of how fast Waldo should be moving.

***Make a new variable* ‘Waldo stride’, choosing ‘For all sprites’**.

We need to make sure it starts off with a sensible value.

***Add* a green-flag script to Waldo**:

Use this stride when moving

We need to change the ‘step one cell’ definition. There are two numbers to think about — how big each move is, and how many moves to do. We know how big one move is, and we know how far we want to move altogether, so we can divide to find out how many moves it will take:

(number of moves) = 40 ÷ (Waldo stride)

You will need something other than ‘40’ if your cells are a different distance apart.

***Change* the ‘repeat’ block in Waldo’s ‘step one cell’ definition script**:

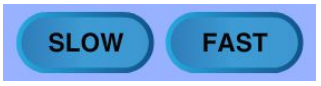
Remember that ‘/’ means ‘divide’.

Experiment with different strides

Test everything still works, then experiment with different numbers for the stride. You have to choose numbers that will divide into 40 evenly: 1, 2, 4, 5, 8, 10, 20, 40. If your cells are some other number apart (not 40), you’ll have to choose from other numbers.

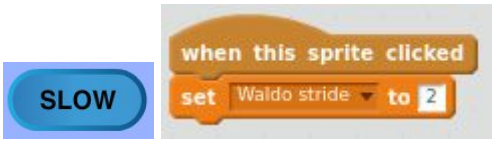
Buttons to change speed

Make two new button sprites and give them costumes saying what they will do. I started with ‘Button2’ but choose what you like.



Each button needs a script so when the player clicks it, Waldo moves quickly or slowly. ‘Quickly’ means a bigger ‘Waldo stride’, and ‘slower’ means a smaller ‘Waldo stride’, so pick two of the allowed numbers.

***Add* one script to each speed-button sprite** to set the Waldo stride size:



Try it!

It doesn’t always work!

If you try to change speed while Waldo is moving, things don’t work properly. Can you work out what is happening?

The problem is that the player might change the ‘Waldo stride’ variable while Waldo is part-way through doing its ‘repeat / move (–) steps’ loop. Let’s say ‘Waldo stride’ is 2. So it will work out that it needs to do 40 ÷ 2 = 20 moves. It starts doing those 20 moves. Part way through doing them, the player might click ‘fast’, which changes ‘Waldo stride’ to 5. So Waldo might do something like 8 moves of size 2 and 12 moves of size 5, and end up in the wrong place.

Sharing variables is difficult

This is often a very tricky thing to get right in programming: When more than one sprite needs to work with the same variable, you have to make sure they agree on how it's used.

Fixing the problem

One way to fix our problem is to make the Waldo sprite the only one which changes the 'Waldo stride' variable, so it can wait until a safe time to change it.

We will use *another* variable to say what the player *wants* 'Waldo stride' to be. Waldo then sets the 'Waldo stride' from the 'Waldo stride wanted' at a safe point.

Make a new variable 

, choosing 'For all sprites'.

***Change* the 'fast' and 'slow' button scripts** so that they set this variable and not 'Waldo stride':

***Add a block* into Waldo's 'step one cell' definition**, just before the 'repeat':

The new block is 'set *Waldo stride* to *Waldo stride wanted*'.

And ***change the variable* which is set in Waldo's 'green flag' script**:

Now only Waldo ever changes the 'Waldo stride' variable, and everything should work.

Challenges

If you get all the above working, here are some challenges to work on, from easiest to hardest:

Show a message if Waldo leaves reactor [quite easy]

We have set it up so Waldo stops if it goes outside the grid of reactor cells. Make the game show a message to the player if this happens.

No adding or moving arrows once Waldo is moving [more difficult]

At the moment, the player can click ‘Go’ and then put more arrows onto the grid, or move the arrows around. This is cheating! Make it so the player can’t add or move arrows unless Waldo is in the ‘reset’ position.

Hint: Use a variable to keep track of whether Waldo is moving.

Make Waldo turn smoothly [quite difficult]

When Waldo hits an arrow, it instantly turns to face the new direction. Make Waldo turn smoothly to face the new direction.

Hint: Depending on Waldo’s current direction and the new direction it should turn to, work out whether Waldo should turn clockwise or anti-clockwise, and then do something like ‘repeat 10: turn 9 degrees’.

Key points

Use ‘when key pressed’ to make testing and development easier.

Break the job down: Work out ‘move one cell’; then work out ‘keeping moving’.

Use ‘stop other scripts in sprite’ block.

Use ‘touching colour (–)’ question block.

Make sprites work together with broadcast messages.

Avoid problems caused by sharing variables between sprites.