

Week 5

Last week we added the 'robot' which follows the instruction arrows round the grid. This week we'll add the 'atoms' the robot has to collect as it walks round the grid.

You can remix [my demo 'Week 4' project](#), or [the version including doing the challenges](#), or start from where you left off last week.

Add an 'atom' sprite

We want to let the level-designer put atoms onto the reactor. An atom will start off exactly like the direction instruction arrows. So, duplicate one of the arrow sprites, give it a sensible name, and choose a costume from the library. Put it in the level-design toolbox near arrows.

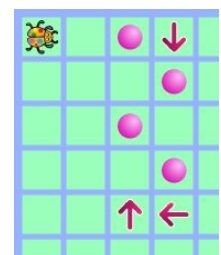


Try it: In full-screen, you should be able to drag atoms onto the reactor grid for Waldo to collect.

Make Waldo collect atoms

If you set up a few atoms and a few arrows, and click your 'go' button, Waldo will completely ignore the atoms. We need to fix this.

It's the atoms which will need to do something — they need to disappear when Waldo goes over them. But it's Waldo that knows when to check — at the end of stepping from one cell to the next.

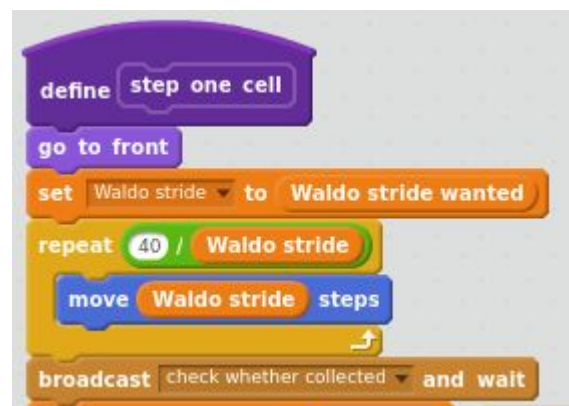


We'll use a broadcast to make the sprites work together. **Add a block into Waldo's 'step one cell' definition script:**

The new block is '*broadcast 'check whether collected' and wait*'. You'll have to choose 'new message...' and type in 'check whether collected'.

I've left out the rest of the script from this picture.

The atom sprite needs to listen out for this message, and check whether that atom should be collected. 'Being collected' just means 'hide'.



Add a script to the 'atom' sprite:



Try it! You should be able to set up a level where Waldo marches round the grid, collecting the atoms.



Reset atoms

If the player doesn't get their arrow instructions right first time, we need to make them re-appear when the player resets Waldo ready to have another go at the level. There's

already a message being broadcast when the player resets Waldo, so we can make the atom listen out for this same message.

Add another script to the 'atom' sprite:



Try it in full-screen — any collected atoms should re-appear when you reset Waldo with the 'stop' button.

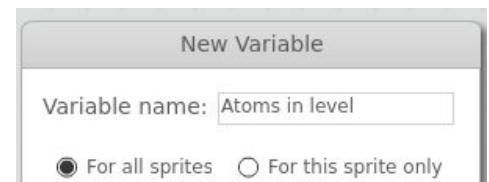


How many atoms are there on the level?

We need to know when Waldo has collected all the atoms. To do that, we need to keep track of many atoms there are, as the person designing the level adds new atoms to the level or removes atoms from the level.

Make a new 'for all sprites' variable to keep track of this:

As usual with variables, we need to think about what value it should start off with, and when it should change.



At the very start, there are no atoms

Add a green-flag script to the atom sprite:



When a new atom clone is made, that's one more atom

Add a block to the atom's 'when I start as a clone' script:

The 'change 'Atoms in level' by 1' is new.



When an atom is deleted, that's one fewer atom

Scratch doesn't have a 'change downwards', but we can use the *negative number* '-1' to make this work.

Add a block to the atom's 'do drag and drop' definition script, inside the 'if touching background colour' at the end, just before 'delete this clone':



Try it

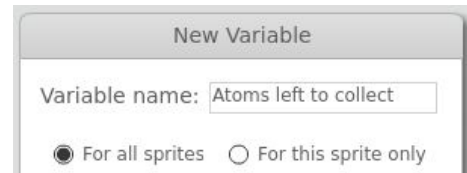
If you tick the 'Atoms in level' variable, to show it on the stage, you should see it count up and down as you add and remove atoms from the level.

How many atoms left to collect while *playing* the level?

Once Waldo starts marching round the reactor, collecting atoms, we need to know when it's collected them all. We need to track this number separately from the number of atoms which exist in the level.

Make a new 'for all sprites' variable:

When should this variable be changed?



All atoms need collecting as Waldo starts

When Waldo sets off, all the atoms in the level need collecting. Also, it will be useful for the player to see how they're doing, so let's show the variable when Waldo is running.

Add two blocks to the top of Waldo's 'when I receive 'start Waldo'' script:

(I've snipped the rest of the script out of this picture.)



When an atom is collected, that's one fewer left to collect

We'll use the *negative number* '-1' to make this work, just like we did for counting how many atoms exist in the level.

Add a block into the atom's 'check whether collected' script:

(The 'change 'Atoms left to collect' by -1' block is new.)



Try it



Go to full screen.

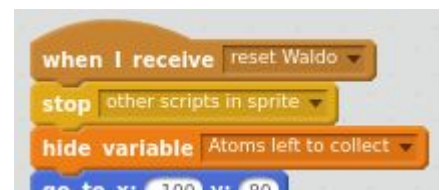
If you set up a level where Waldo collects some atoms, you should see the 'Atoms left to collect' variable counting down as Waldo goes over each atom and collects it.

Hide variable when Waldo not running

To be tidy, we should hide the 'Atoms left to collect' variable when it's not useful to the player.

Add a block to Waldo's 'reset Waldo' script:

The 'hide variable' block is new.



Player wins if all atoms collected

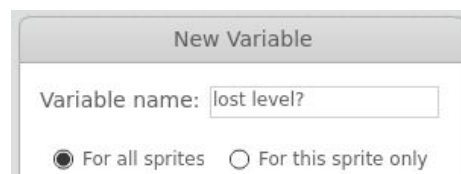
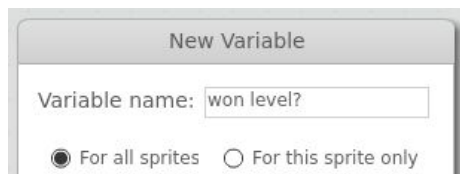
Up until now, there is only one way that Waldo stops moving: If it crashes out of the reactor. Now there are two ways:

- Waldo crashes out of the reactor — this means the player has lost the level.
- Waldo collects all the atoms — this means the player has won the level.

Waldo needs to keep going until either the player wins or the player loses.

To make it clear what our program is doing, we'll create one variable to keep track of whether the player has won, and another to keep track of whether the player has lost.

Make two new 'for all sprites' variables, 'won level?' and 'lost level?':



These variables should start off at 'no' when Waldo starts.

Add two blocks into the top of Waldo's 'start Waldo' script:

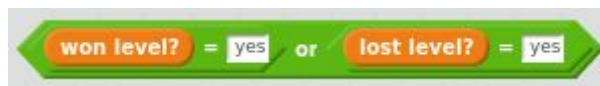
The two 'set 'won/lost level?' to no' blocks are new.



Use these variables in Waldo's main loop

The main loop in Waldo's 'start Waldo' script needs changing to use these variables. There are a few changes in this section of the script.

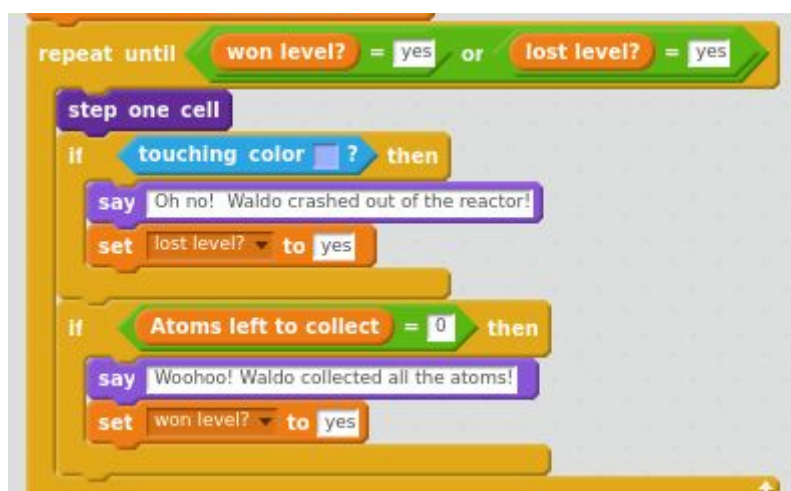
The question we ask to know when to stop now needs to be



And we need to check for winning, and also for losing, and set the right variable if the player has won or lost. We can also show a message.

To put this all together, we need to make quite a few changes to the 'repeat until' loop.

Change the 'repeat until' loop in Waldo's 'start Waldo' script to this:



Try it!



In full-screen, design a level with a few atoms and check you can set up instruction arrows to make Waldo either win or lose.

Make 'blocked' cells

The game is still quite easy. We need a way for the 'designer' to block off some cells.

We can do this by duplicating an arrow sprite, and changing its costume into something meaning 'not allowed in this cell'. I chose the cross shown (it's the library costume 'button5-b'). Give it a useful name, adjust its size if needed, and put it by the other tools (arrows and the atom).



You can now put these blockers into cells, but Waldo will just march right over them. We need to make Waldo do something if it touches a cell blocker. At the moment, Waldo checks for 'marched out of reactor' and 'collected all atoms'. We need to add a check for 'landed on blocked cell', which will lose the level.

Add an 'if / then' block to the 'repeat until' loop in Waldo's 'start Waldo' script:



Challenges



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

Have more than one sort of atom [easy]

Make there be other colours of atoms. Should they all work the same? Or, for a more difficult challenge: can you think of ways they could behave differently?

Play a sound when Waldo collects an atom [quite easy]

Sprites can have sounds. They start off with a 'pop' noise, which would work well for the

'collected' noise. You'll need the  block from the  section, and you'll need to find a good time for the sound to happen.

Make 'game over' screens after winning or losing [medium]

Instead of Waldo just using 'say' to tell the player whether they've won or lost, make whole-screen announcements.

Separate 'level design' tools from 'play level' tools [harder]

The 'level designer' should be placing the atoms, and the 'level player' should place arrows, but at the moment arrows and atoms are all the same.

Divide the tools into 'designer' and 'player', and put in a way to switch between 'design' and 'play' modes.

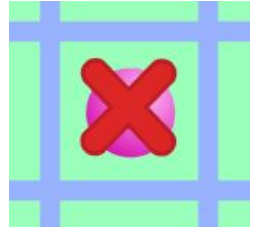
Stop arrows or atoms being in the same cell [lots harder]

If you have an atom in a cell, you can drop a blocker onto the same cell, which makes no sense. Work out how to stop this happening. There are two ways this could work:

- When you drop the blocker on the atom, the blocker is deleted.
- When you drop the blocker on the atom, the atom is deleted.

I think the second is probably better but you could argue both ways.

Even harder is to stop the player dropping an arrow on top of an atom or blocker!



Have some more ideas!

Key points

Use 'hide' to make it look like something is being collected; use 'show' to bring it back.

Counting how many clones have been made: +1 when one created, -1 when one deleted.

Use broadcasts to make sprites work together.

Counting how many atoms are left to collect while playing the game with a separate variable.

Use 'show variable' and 'hide variable'.

Moving the 'level lost?' check into a variable; adding a 'level won?' variable.

Adding a 'blocker', and including it in the 'level lost?' check.