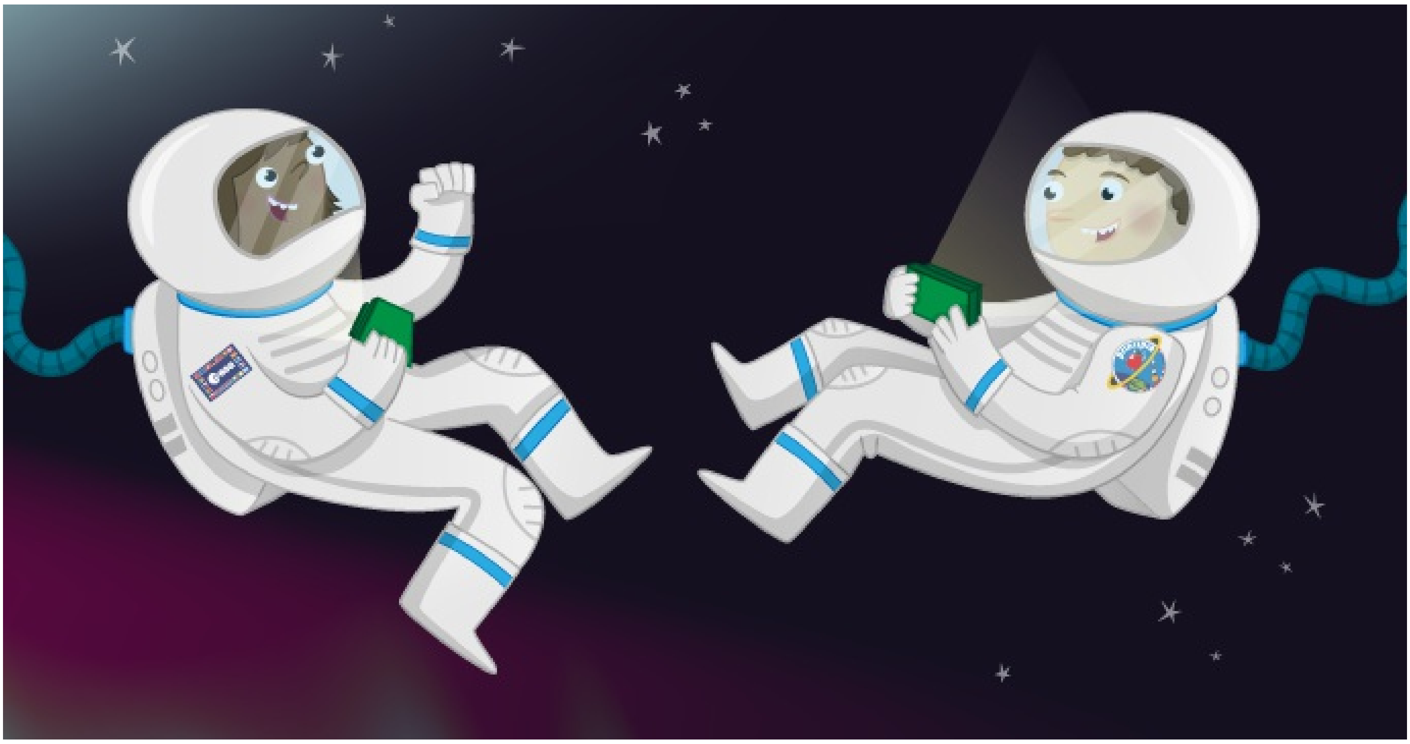
Astronaut Reaction Times Game



Things happen quickly when you’re traveling at 16,000 miles per hour (around 7,000 metres per second), and when debris and micro meteoroids are heading towards you at around 22,500 miles per hour. Quick reactions and a steady hand are also needed for tasks requiring fine motor skills, such as controlling robotic arms. Astronauts are trained intensively to speed up their reactions to incidents, and to prepare them for all eventualities.

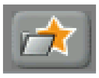
Let's create a game in Scratch to test your reaction skills, and those of your friends and family, to see if you could become an astronaut like Tim Peake.

Set the stage with a space theme

1. Open up ScratchGPIO7 plus on the desktop. If it is missing, double click InstallScratchGPIO.
2. Create a new file by selecting **File** and **New**.
3. Delete the Scratch Cat sprite by right-clicking on it and selecting **Delete** from the menu that is displayed.
4. For this project, you need a space-themed background and an Astronaut sprite. To add a background in Scratch, click on `stage` in the sprites palette and then click on `Backgrounds` next to the `scripts` tab.
5. Click on `Paint` to draw your own background or `import` to use the same images as this resource. Connect your Raspberry Pi to the internet, and download this Space

background files/Space-background.png and British ESA astronaut Tim Peake sprite files/Astronaut-Tim.png. Save them somewhere that you will be able to find them on your Raspberry Pi. You can also find more images to use here https://github.com/raspberrypilearning/astronaut-reaction-times/tree/master/files if you do not like those.

6. Next, add a new sprite by clicking on the `import a new sprite` icon on the sprites palette (which looks like the image below), selecting `Astronaut-Tim` from the choices and clicking **OK**.

7. Save your Scratch project work by clicking on **File** and **Save As**. Name your program **Astronaut Reaction Game** and save it in your home directory or some place that you can find it later.

Create variables to store data

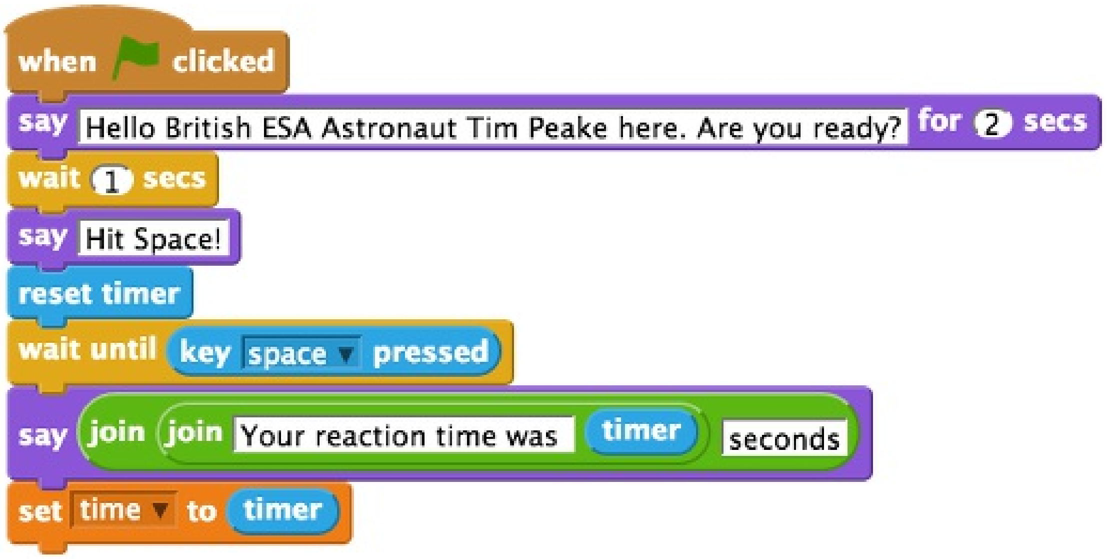
Variables are simply a temporary place to store some information, for example a reaction time or a name.

1. To create a variable, click on `Variables` in the blocks palette and then click `Make a Variable`. The New Variable window opens and asks you to type a name for your variable.
2. Name the first variable `time` and ensure that `For all sprites` is checked before clicking **OK**. A variable holds a value that can be changed. The time variable you have created is an example of a value that can be changed and used inside different scripts. You will use it to store the reaction times of players.
3. You'll see some orange blocks are added to your `Variable` area called `time`, and a small counter box will appear on the stage.

Begin the reaction game script

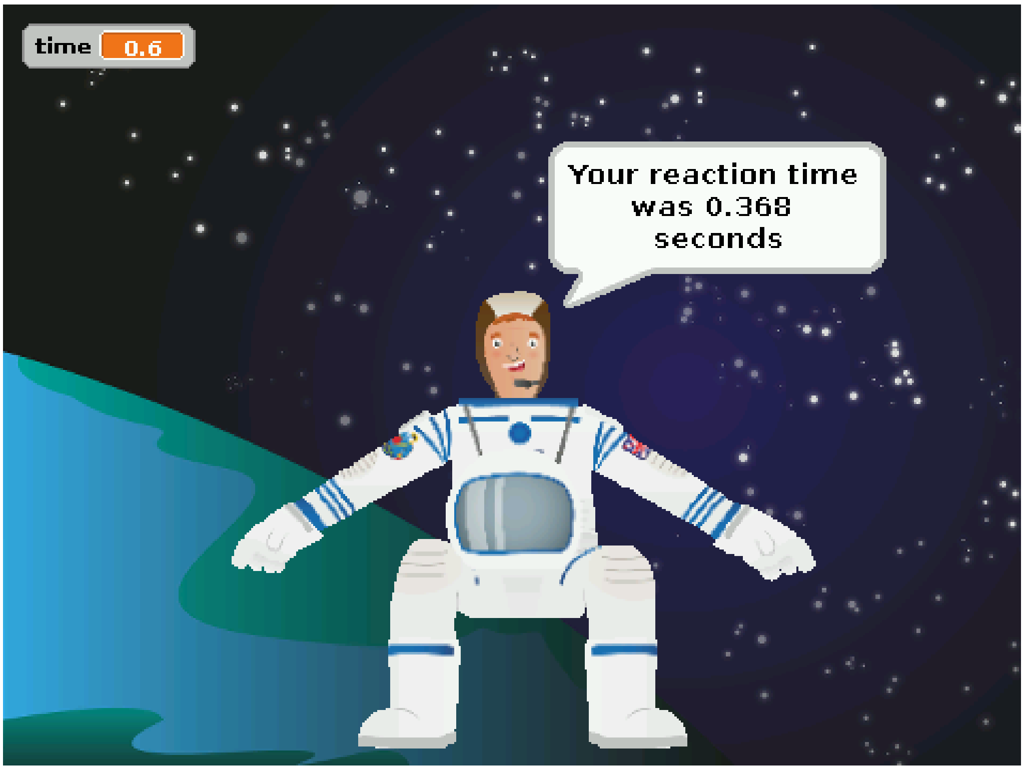
1. Click on your Sprite to select it in the sprites palette.
2. Select the `When green flag is clicked` control block from the blocks palette and place it onto the scripts area.
3. Then click on `Looks` and connect the `say for 2 secs` block to the first control block on the scripts area. Amend the text to say `Hello! British ESA Astronaut Tim Peake here. Are you ready?`.

4. Add a `wait 1 secs` block underneath.
5. Connect another `say` block and change the text to `Hit Space!` .
6. Click on `Sensing` and connect the `reset timer` block. This will set the timer to 0 so that you will get an accurate measurement of how long it takes for someone to hit the space bar.
7. Use the `control` block `wait until` and place a `key space pressed?` sensing block inside the white space of the wait until block. This will pause the program until the player presses the space bar.
8. Then connect another `say` block. Once the space bar has been pressed, you want to display the reaction to the player. To do this, you need to place an `operators` block called `join hello world` inside the white space in the say block. Replace the word `world` with the word `seconds` .
9. You will then need to replace the word `Hello` with another `join hello world` operators block, replacing the `Hello` text with `Your reaction time was` and the `world` text with the `timer` sensing block.
10. Finally, select the `set time to` block from the variables section and add it to your script. Place the `timer` sensing block inside where it reads `0` .



```
when [flag] clicked
say [Hello British ESA Astronaut Tim Peake here. Are you ready?] for (2) secs
wait (1) secs
say [Hit Space!]
reset timer
wait until <key [space] pressed>
say (join (join [Your reaction time was] (timer)) [seconds])
set [time] to (timer)
```

11. Save your game and test it works by clicking on the green flag. When Tim says "Hit Space!", press the space bar. Your time should be displayed like this:

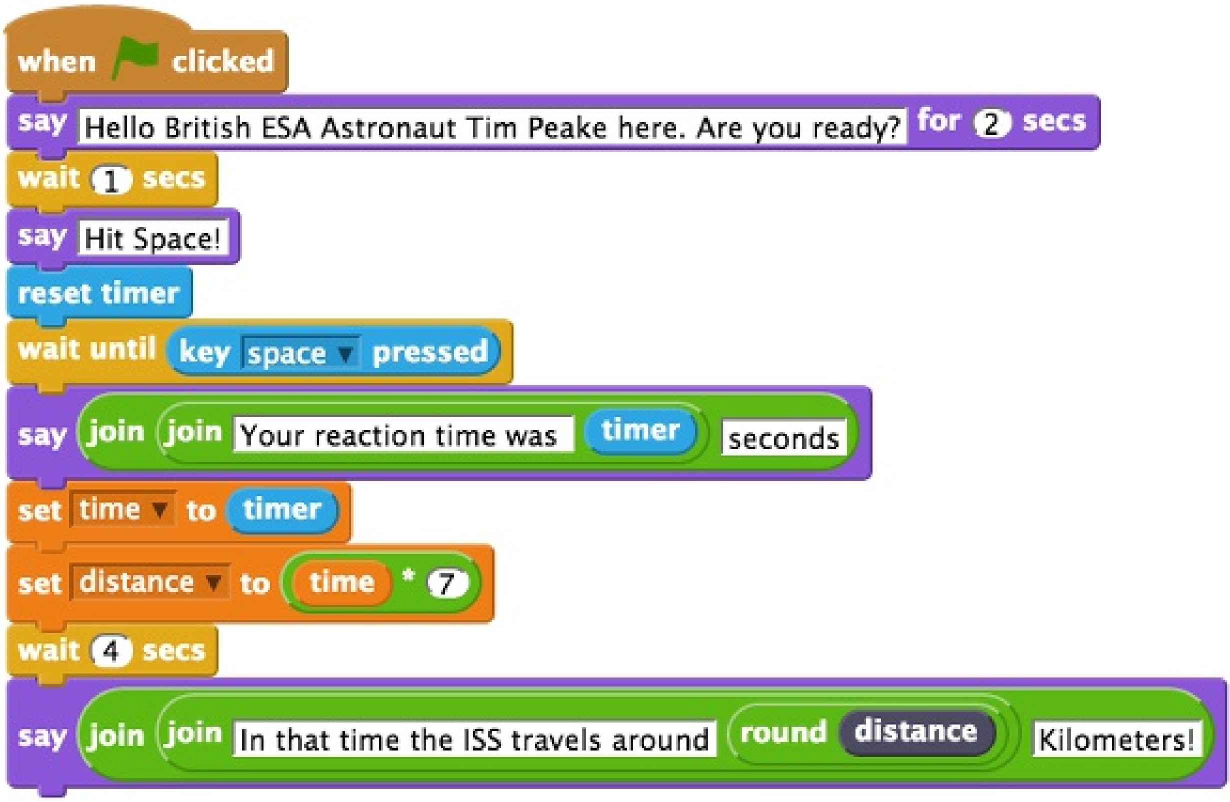


Comparing player's reaction time to the ISS orbit

If you are happy with your reaction game and have tested that it works, then you can move onto adding to the script to compare the player's reaction time to how fast the ISS is travelling, to calculate how far it would travel in that time.

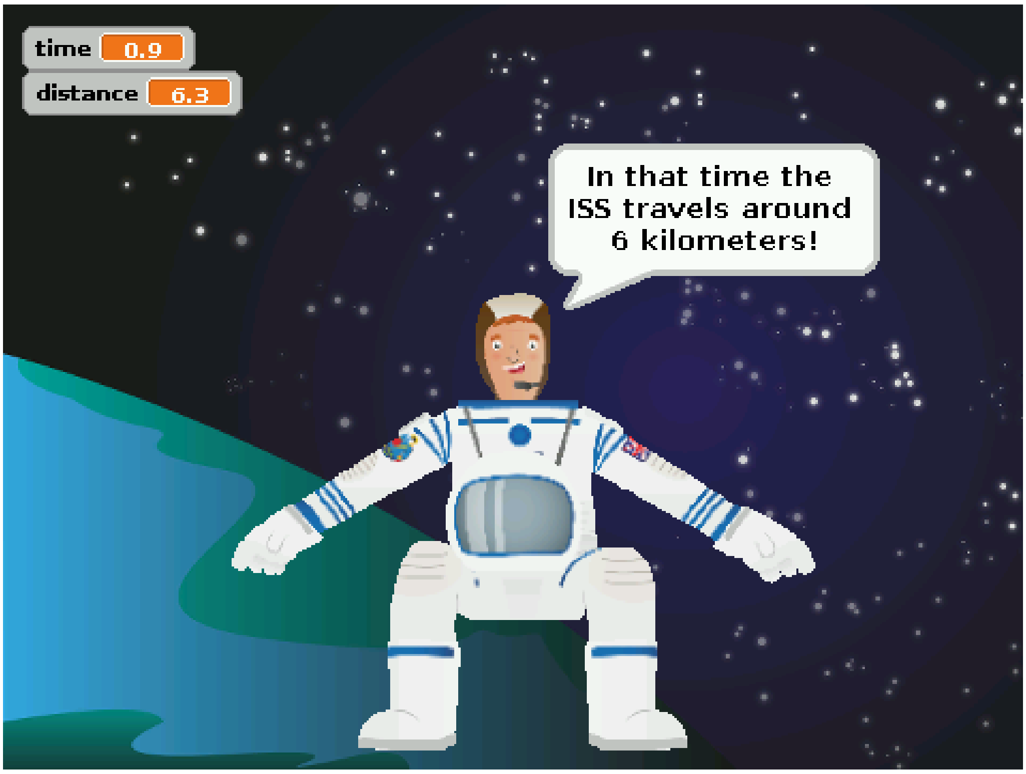
1. First you will need to make a new variable called `distance` in the same way you did earlier.
2. Attach a `set distance to` variable block to your script. Place an `operators` multiply block `0*0` inside where it reads `0`. To calculate the distance travelled by the ISS you need to take the player's reaction time, which is stored in the time variable, and multiply it by 7. This is because on average the ISS travels 7 kilometres per second!
3. Add the `time` variable block into the right hand side of the multiplying operator and type `7` in the other side, so that the whole block reads `set distance to time * 7`.
4. Next, add a `wait 4 seconds` control block.
5. Then add a `say` block. As in the previous step, place a `join hello world` block inside. Replace `World` with `kilometres`. Insert another `join Hello World` block to replace `Hello`. Replace the `Hello` text in this new Join block with the text `In that time the ISS travels around`. Then

replace `World` with a `round` operator block and fill the white space with the `distance` variable block like this:



```
when ⚑ clicked
say [Hello British ESA Astronaut Tim Peake here. Are you ready?] for (2) secs
wait (1) secs
say [Hit Space!]
reset timer
wait until <key [space ▾] pressed>
say (join (join [Your reaction time was] (timer)) [seconds])
set [time ▾] to (timer)
set [distance ▾] to ((time) * (7))
wait (4) secs
say (join (join [In that time the ISS travels around] (round (distance))) [Kilometers!])
```

6. Save your game and test that it works by clicking on the green flag.



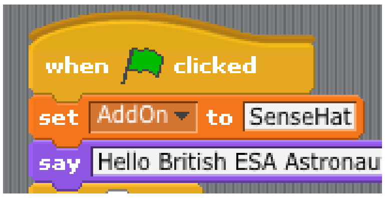
time 0.9
distance 6.3

In that time the ISS travels around 6 kilometers!

Sense HAT

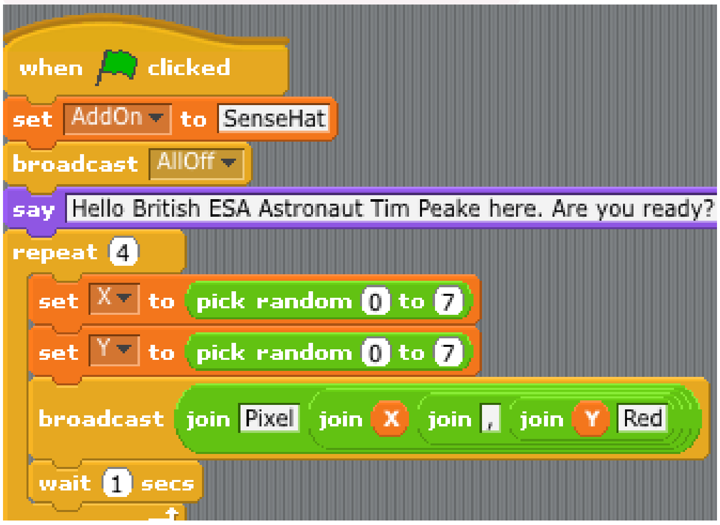
So now we have our basic game made, lets make things a little more difficult for the player, lets use the LED Matrix on the Sense HAT.
To make the game more difficult, how about we make a random number of red stars appear on the Sense HAT followed by a purple star. The player must hit space when they see the purple star.

1. First lets make sure we let ScratchGPIO know we want to use the Sense Hat by creating a variable called `AddOn` and setting it to `SenseHat` at the start of our program.

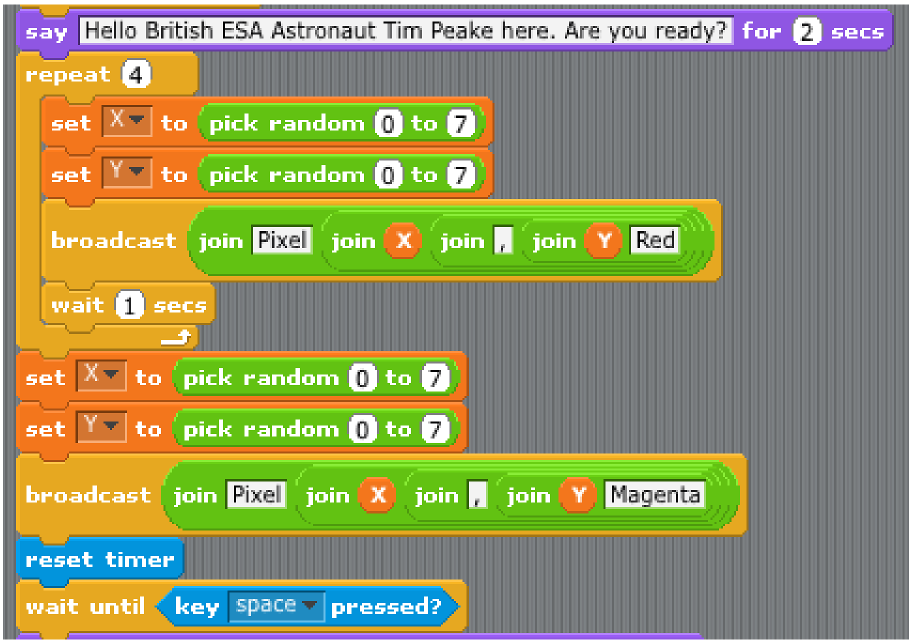
   ```
   when ⚑ clicked
   set AddOn to SenseHat
   say Hello British ESA Astronau
   ```

2. Make sure the LED Matrix is clear by broadcasting (using broadcast block from control) the word `AllOff`.
3. Lets create 2 new variables, one called `X` and the other called `Y`. These will store the x and y coordinate of a new random pixel.
4. Replace the `Wait 1 second` block with a repeat loop which inside contains a `Broadcast Pixel X.Y Red` inside it.

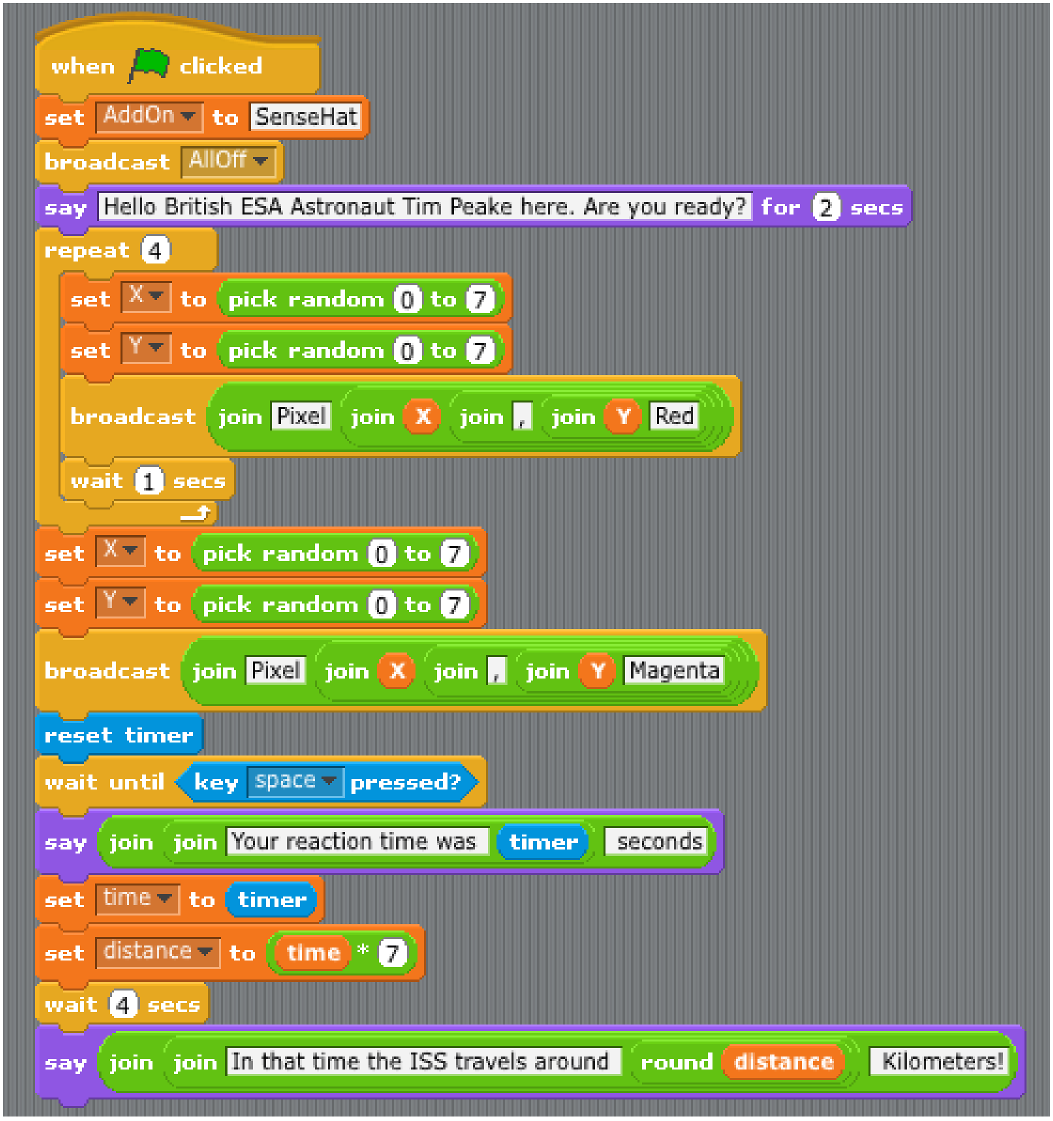
   ```
   when ⚑ clicked
   set AddOn to SenseHat
   broadcast AllOff
   say Hello British ESA Astronaut Tim Peake here. Are you ready?
   repeat 4
       set X to pick random 0 to 7
       set Y to pick random 0 to 7
       broadcast join Pixel join X join , join Y Red
       wait 1 secs
   ```

5. We now have the 4 random red stars appearing, so now we need the purple star. For that we just need to add nearly exactly same code below it, just will need to change the colour from Red to Magenta. Make sure you don't put it inside the repeat loop!



```
say Hello British ESA Astronaut Tim Peake here. Are you ready? for 2 secs
repeat 4
    set X to pick random 0 to 7
    set Y to pick random 0 to 7
    broadcast join Pixel join X join , join Y Red
    wait 1 secs
set X to pick random 0 to 7
set Y to pick random 0 to 7
broadcast join Pixel join X join , join Y Magenta
reset timer
wait until key space pressed?
```

6. The final program now looks like this



```
when clicked
set AddOn to SenseHat
broadcast AllOff
say Hello British ESA Astronaut Tim Peake here. Are you ready? for 2 secs
repeat 4
    set X to pick random 0 to 7
    set Y to pick random 0 to 7
    broadcast join Pixel join X join , join Y Red
    wait 1 secs
set X to pick random 0 to 7
set Y to pick random 0 to 7
broadcast join Pixel join X join , join Y Magenta
reset timer
wait until key space pressed?
say join join Your reaction time was timer seconds
set time to timer
set distance to time * 7
wait 4 secs
say join join In that time the ISS travels around round distance Kilometers!
```

Challenges

- You could make the game more interactive by getting the Tim Peake sprite to ask for the player's name before they start and storing that information in a new `name` variable. Then you would be able to personalise the `say` blocks to include the player's name.
- Using some of the other images, could you change the background and sprites to make the game more interesting?
- Using the `pick random` block, could you randomise how many red stars appear before the purple star?
- The joystick on the Sense Hat is set up as a virtual keyboard. When you press up with it, the up arrow key it detected by the Raspberry Pi. Could you use that instead of the space key?
- Could you make the stars more than a single pixel? Perhaps exploding outwards by 1 pixel?

Credits

This resource was created by Andrew Mulholland and is based on the Raspberry Pi Foundation resource, "Astronaut Reaction Times Game" https://www.raspberrypi.org/learning/astronaut-reaction-times/