Scratch Olympics Hurdler

A hurdler game in Scratch

Scratch

Step 1 What you will make

In this resource you will use Scratch to make a hurdling game, which requires the player to rapidly hit the keyboard to make the hurdler run, and use expert timing to make them jump at the right time.

What you will learn

By creating a Scratch Olympics Hurdler game, you will learn:

- How to animate costumes in Scratch
- How to use conditional selection and Boolean operators to make changes in a program
- How to use conditional selection within loops
- How to use variables to model speed, distance, and time

This resource covers elements from the following strands of the **Raspberry Pi Digital Making Curriculum** (https://www.raspberrypi.org/curriculum/):

- **Combine programming constructs to solve a problem** (https://www.raspberrypi.org/curriculum/programming/builder)

Step 2 What you will need

Hardware

- An internet connected computer

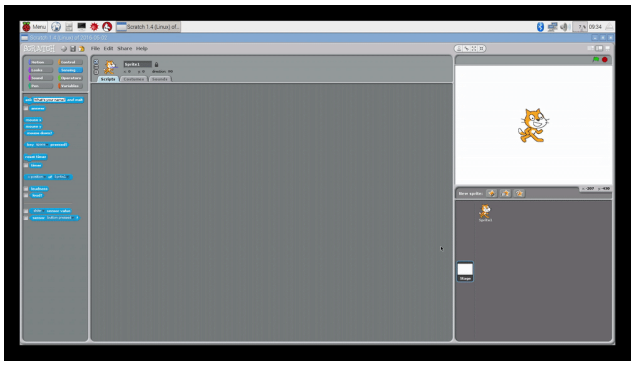
Software

- Scratch 1.4

Step 3 Getting hold of the sprites

- Download this **zip file** (https://projects-static.raspberrypi.org/projects/scratch-olympics-hurdler/757ed28521a5af71a144903d1ed89d0570d787ac/en/images/assets.zip) to your home directory and unzip its contents.
- Have a look inside the directory; you should see 4 directories called `background`, `items`, `misc`, `runner`, and `turtle`.

Step 4 Setting up the assets



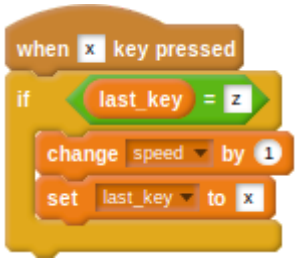
- Open Scratch by clicking on `Menu` > `Programming` > `Scratch`.
- Now, click on the `background` icon and import the new background from the `assets` directory. You can then delete the old background.
- Click on the icon to import a new sprite and then choose the `run-1` image. Next, import `run-2`, `run-3`, and `run-4` as additional costumes. You can then delete the old cat sprite.

Step 5 Capturing the key mashing

- The first step is to capture the `x` and `z` key presses, and use the speed at which the player is pushing the keys to control the size of a variable. To do this you'll need a variable that stores the last known key press. Create a variable called `last_key` and set it to `z` when the green flag is clicked.

- For the next script you'll need a new variable called `speed`, so go ahead and create it now. It can be set to `0` when the game begins.

- When the `x` key is pressed, if the `last_key` is equal to `z`, then the `speed` variable can be increased and the `last_key` can be set to `x`. This will ensure that the player can't cheat and keep hitting the `x` key to make the speed increase.

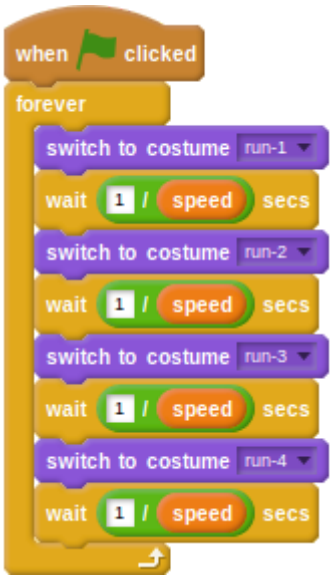


- The same can be done for the `z` key. In combination, these two scripts force the player to hit the keys *alternately* in order to increase the speed variable.

- Now test your script. Click the green flag, then repeatedly press the `x` and `z` keys and watch the speed variable increase.

Step 6 Animating the hurdler

- At the moment, the hurdler has 4 costumes as part of what's called a *walk cycle* (or run cycle in this case). When these costumes are switched, the character appears to run on the spot. The time delay between costume switches should depend on the `speed` variable. The higher the `speed`, the quicker the costume change should be and therefore the smaller the delay. You can get this effect by dividing `1` by the `speed` variable to calculate a delay.

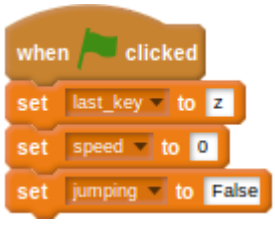


- If you run this script as it is, you'll get an error, because `speed` starts off with a value of `0`. This means the computer is trying to calculate `1 / 0`, which it can't do. It's a very common error that programmers make in their code. To fix this, you can use a conditional to make sure that the calculation only occurs when `speed` is larger than `0`.

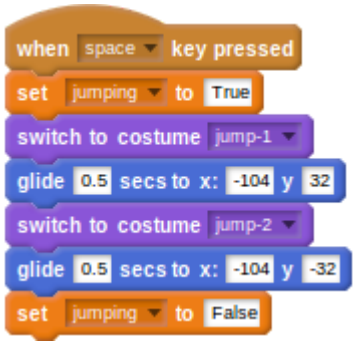
- Now you should be able to test your script and watch the hurdler running on the spot as you press the `x` and `z` keys.

Step 7 Jumping

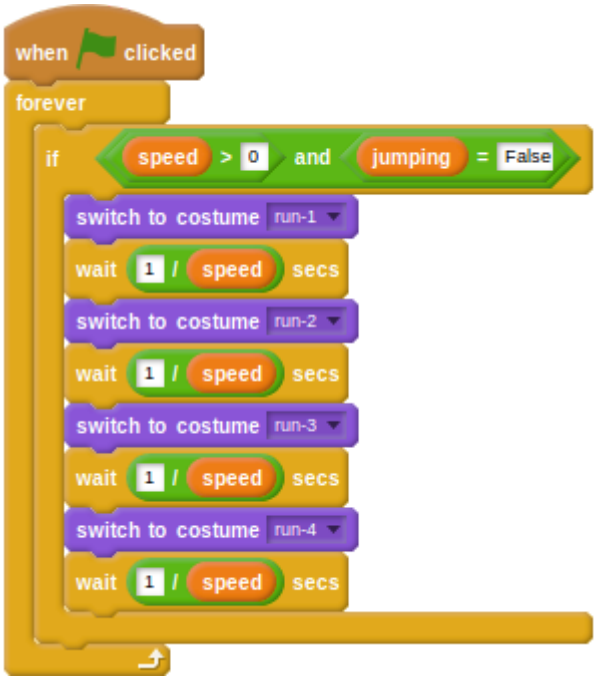
- Hurdlers need to jump. You'll need a few more costumes for this part, so look in the runner directory and import the `jump-1` and `jump-2` costumes for your hurdler.
- You'll need a new variable for this part called `jumping`. This is because other scripts will need to know when the character is jumping. Create the new variable and set it to `False`.



- The character should jump when the space bar is pressed. The first thing that happens is the `jumping` variable should be set to `True`, then the costume can be changed to `jump-1` and the character can glide upwards. Next, the costume can be changed to `jump-2` and the character can glide back down again. Finally, the `jumping` variable can be returned to `False` to indicate that the jumping animation has finished.



- Test your script; it might surprise you to see that the character's costume doesn't change. This is because the walk cycle you set up previously is still working. You'll need to stop this walk cycle when the character is jumping. To do this, you can use an `and` conditional operator to check that both `speed > 0` and `jumping = False` for the walk cycle to work.



- Now have a go and you should find your character jumps when the space key is pressed.

Step 8 Slowing down

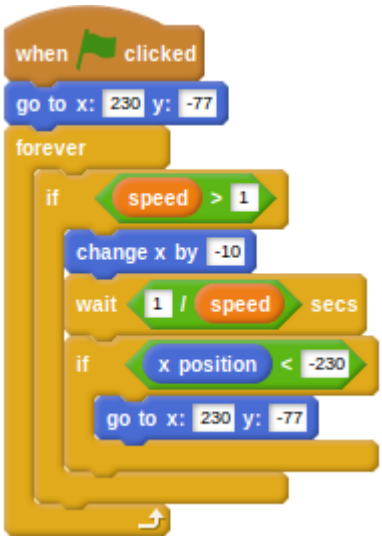
- At the moment, the more you press the `x` and `z` keys, the faster the character runs. There needs to be a way of slowing the hurdler down, so she doesn't win too easily. This can be done on your initial script

that sets the starting variables. You just need to add an infinite loop that will check if the speed is greater than 1, and then lower it every few 100ths of a second.

Step 9 Adding in hurdles

For the final part of this worksheet, you can add in hurdles that the character will have to jump over.

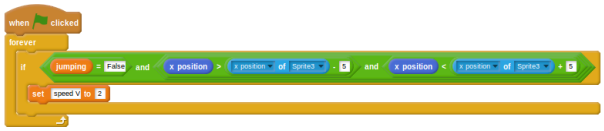
- Import the hurdle.png sprite from the `assets/items` directory.
- This sprite needs to begin at the far right of the screen, then it should continually move left across the screen at a pace that's proportional to the speed of the character. When it hits the far left of the screen, it should instantly appear on the right again.



Step 10 Making the hurdles an obstacle

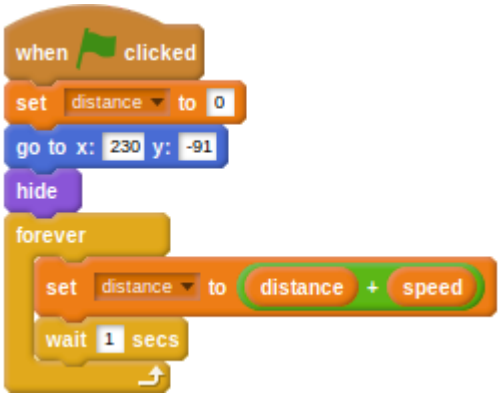
At the moment the runner can just plough straight through the hurdles. She needs to be slowed down if she doesn't jump.

- Back on the hurdler sprite, add in a new `when green flag clicked` block.
- This next part is a little complicated. The runner should be slowed down if she:
  - Isn't jumping
  - Has an x position just before the hurdle
  - Has an x position just after the hurdle
- This can be achieved using two `and` logical operators, checking if:
  - `jumping = False`
  - `x position > x position of hurdle - 5`
  - `x position < x position of hurdle + 5`
- If all those conditions are met, then she must have hit the hurdle and her speed can be dropped.



Step 11 Making an end to the game

- To finish off a completed game, you need to add in a finishing line. You can find one in the `assets/items` directory.
- Import this as a new sprite into your Scratch game, and approximately position it into the runner's lane.
- To start off, you need use a variable to control how far the hurdler has to run. Create a new variable and call it `distance`.
- The first script to be added to the finish line will set `distance` to `0` when the game begins, position the finish line to the far right of the screen, and hide it. Next, `distance` has to be increased by the `speed` of the runner every second.



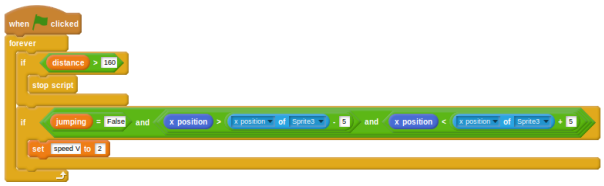
- Now that the finish line is ready to go, you can make it appear when the value of `distance` hits whatever value you desire (200 in the example below). It can then begin to move across the screen towards the hurdler. When the hurdler touches the finish line, all the game scripts should end.

Step 12 Hiding the hurdles

You may have noticed that the hurdles stay on the screen, even when the player is approaching the finishing line. A little change to some of the scripts, and the game can be made more realistic.

- The first thing to do is to hide the hurdles when the `distance` variable climbs above a certain value. Click on the hurdles sprite you have imported and then alter the script to include a conditional to check for this. You'll also need to add in a `show` when the script is started.

- Next, the script that slows the player down when they hit the hurdles needs to be stopped at the same `distance`, so the player can't run into invisible hurdles. Click on the hurdler sprite and edit the collision script.



- Have a play of the game and make sure that the hurdles disappear towards the end. You might need to tweak the variables a little to get the perfect result.

Step 13 Breaking the finish line

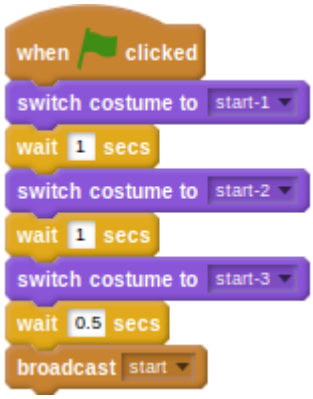
You may have noticed a second finish line graphic in the `items` directory, showing a broken finish line. You can use this in your game.

- Click on the finish line sprite and then click on the *Costumes* tab.
- Now click on *Import* and choose the `finishline-broke` graphic from the `items` folder in `assets`.
- Back on the *Scripts* tab, edit the script that makes the finish line appear. You're going to add a *conditional* so that when the finish line is touched by the player, it breaks and the player continues running for a little while.

Step 14 Setting up the start

There are three graphics in the `runner` directory that you have not yet used. These are called `start-1`, `start-2`, and `start-3`. You can use these at the beginning of the game, to start the player off.

- Click on the hurdler sprite and then *Costumes*.
- Now import the three *starting* costumes for the hurdler.
- Click back on the *Scripts* tab.
- Now you can add in a new script to start the game. When the `green flag` is clicked, the starting costumes can be animated, before the script `broadcasts` start to indicate the game can begin.



- Now you need to edit the main animation loop so that it starts on the broadcast, rather than on the green flag being clicked.

Step 15 What next?

In the `misc` directory are lots of objects you can have a play around with. Each can be positioned somewhere on the stage and so long as they move from left to right at the same speed as the hurdles, then they'll help add to the illusion of movement. This section is left up to you. You could make each one an individual sprite and add them in, or perhaps make them costumes of a few sprites which appear randomly! It's really up to you. Have a look at the GIF below to see the effect you can achieve. Don't forget the turtle graphic either, in the `turtle` directory. What could you make him do in the race?