



SWEET SHOP REACTION GAME

Some penny sweets or candy can make great input devices for a game using a Raspberry Pi. Let's turn a squidgy sweet into an input button for your Raspberry Pi. You will then create a quick reaction game in Scratch that counts how many times in 10 seconds a player can squeeze your sweet input device!



CREATE A SWEET MUNCHING SPRITE

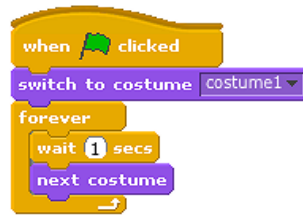
The sweet shop reaction game needs a munching face to entertain the player. You will draw a face using the paint editor in ScratchGPIO, and animate it to open and close its mouth.

1. On the desktop of your Raspberry Pi you should see a **ScratchGPIO** icon. To open it, double-click the icon. It is very important that you use this version of Scratch and not the default application. Click **Ok** to enable remote sensor connections.



2. Delete the Scratch Cat sprite by **right-clicking** on it with your mouse and selecting **delete**.
3. Click on the **paint new sprite** icon above the sprites palette, and draw a face with a mouth that is closed using the paint editor. When you are happy with your sprite click **OK**.
4. Next, with your newly painted sprite selected, click on the **Costumes tab**. Rename the costume to **face1** by clicking on the sprite name followed by the edit button, and typing the new name.
5. Click the **copy** button to make an exact copy of the face. You will now have two identical faces on the costumes tab called face1 and face2.
6. The next step is to edit face2 to change the mouth from closed to open. With face 2 selected, click on the **edit** button to open the paint editor.
7. Erase the mouth using a paintbrush tool or an erase tool and then replace it with an open mouth. When you are happy with your sprite costume click **Ok**.
8. To animate your sprite to switch between costumes, you will need to click on the **Scripts tab** of your sprite. Drag the control block `when green flag clicked` from the blocks palette to the scripts area.
9. Next, drag the look block `switch to costume face1` and connect it to the control block.

10. Then add the control block `forever` underneath the look block.
The `forever` block is a loop that will run the same sequence of blocks inside it over and over again.
11. Add the control block `wait 1 secs` and the look block `next costume` inside the forever loop.
12. Change the time from 1 second to half a second. How could you represent time as a value? If 1 is a whole, what would half of 1 be?
13. Time to save your work so far and test that your script to animate a sprite works. Go to **File** and **Save As**. Name your file **SweetShopGame** and click **Ok**.
14. Finally, click on the **Green Flag** in the top right hand of the screen and you should see your sprite face open and close its mouth.



DESIGN A SWEET SHOP BACKGROUND

To make the game a little more interesting, let's set the scene by changing the background from the default white to something a little more exciting, like a gradient colour or sweets in a shop!

1. Click on the **stage** icon next to the sprites palette to change the background.
2. Next, locate the **backgrounds** tab and select it with your mouse.
3. If you want to draw your own background, click on the **edit** button underneath the **background1** label; this will open the paint editor. Use the drawing tools to make a more interesting and colourful background.

Or

If you would rather add a new background using an image, delete background1 by clicking on the **x** underneath the background1 label. Then click on the **import** button to use a background from the image library, or from a picture that you have saved.

4. When you are happy with your background, click on **sprite1** in the sprites palette, ready to program the reaction game mechanics in the next step.



PROGRAM THE SWEET SHOP REACTION GAME MECHANICS

Many people enjoy testing their reaction time against a clock. Let's create a reaction game using ScratchGPIO that we can connect a squidgy sweet button to later on. The object of the game is to see how many times you can squeeze the sweet button in 10 seconds.

You will need to create two variables for this game: one to count the button presses and one to count

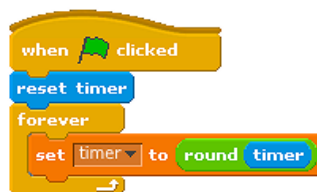
time.

1. Click on **variables** from the blocks palette and select `make a variable`. Name the first variable **counter** and click **ok**.
2. Repeat the first step to create another variable named **timer**.
3. Click on the control blocks palette and drag the `when green flag clicked` block on to the scripts tab of your face sprite.
4. Next, add the variable block `set counter to 0` so that at the start of each game the counter is reset to 0, ready to test the player's button-pushing skills.
5. Add a `forever` looping block, connect it and then place an `if` block inside the `forever` block.
6. There is a small blank space on the `if` block; this is so that you can add other blocks. In this space you need to first add the **operator** block `=`.
7. Blocks can be added on either side of the `=` block. On the left hand side add the sensing block `pin3 sensor value` and on the right hand side type the value `0`. Using the drop-down menu change **slider** to **pin3**.
8. Inside the `if` block add the variable block `change counter by 1` and `play drum 48 for 0.2 beats`. You can select any drum noise that you like from the drop down menu.



To set a time limit for the game that counts upwards, you need to add two further scripts:

1. Add `when green flag clicked` block to the scripts and connect the sensing block `reset timer` to it.
2. Underneath, connect a `forever` looping control block.
3. Inside the loop add the variable block `set control to 0` and using the drop-down menu on the block, change **control** to **timer** so that the block reads `set timer to 0`.
4. Replace the value `0` in the `set timer to 0` block with the operator block `round`.
5. Then add the sensing block `timer` inside the space on the 'round' block. The completed block should look like:



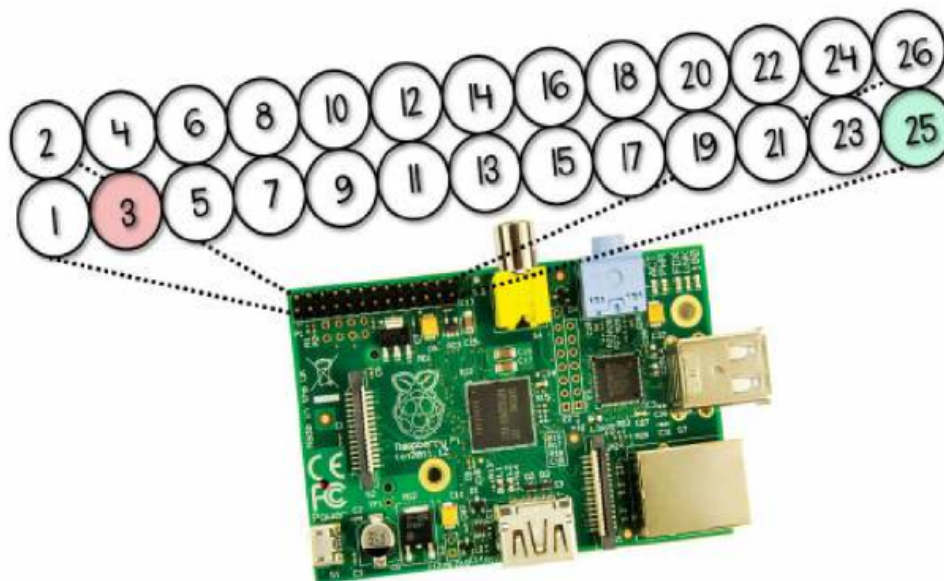
6. Add another `when green flag clicked` control block to the scripts area and connect a `wait until` block to it.
7. Add the operators block `=` to the space in the `wait until` block. In the left hand space add the variable block `timer` and on the right hand side type a value to represent time. If you want your game to last for 10 seconds then type `10`.
8. Connect a `stop all` control block to the end of this script.
9. Finally, save your game by clicking on the save icon at the top of the screen.



WIRE UP YOUR SWEET BUTTON

You will need to connect a sweet or piece of candy to your Raspberry Pi to act as an input device and test it.

1. Take the metal paper clips and unfold them to make straight wires.
2. Insert the paper clip wire or dress pins into the end of a female to female jumper cable.
3. Do the same to the other jumper cable so that the two cables are identical.
4. Insert the paper clips into a soft sweet so that they are close to each other but not touching.



Raspberry Pi GPIO header pins. The diagram above the pins shows the pin numbers. You will be using pin 3 and pin 25.

5. Take the other end of one of the jumper cables (not connected to a paper clip) and push onto pin 3 of the General Purpose Input-Output (GPIO) header, which is connected to one of the GPIO channels.
6. Take the end of the other jumper cable and push onto pin 25 of the GPIO header, which is connected to ground.

Warning! You can damage your Raspberry Pi if you do not use the GPIO pins correctly!

Your sweet input device is not a real button and will not give accurate results for your game. It is just a bit of fun.



PUT IT ALL IN A BOX!

Congratulations on making your Sweet Shop Reaction Game. If you have time why not make and decorate a box to put the Raspberry Pi and cables into?



WHAT'S NEXT?

- Can you add user input to your scratch program to find out the name of the player?
- Can you connect a real physical button instead of a sweet?
- Can you write a similar program using a text based programming language like Python?



This learning resource is provided for free by the [Raspberry Pi Foundation](https://www.raspberrypi.org/) under a [Creative Commons](https://creativecommons.org/licenses/by-sa/4.0/) licence. Find more at [raspberrypi.org/resources](https://www.raspberrypi.org/resources) and github.com/raspberrypilearning.

