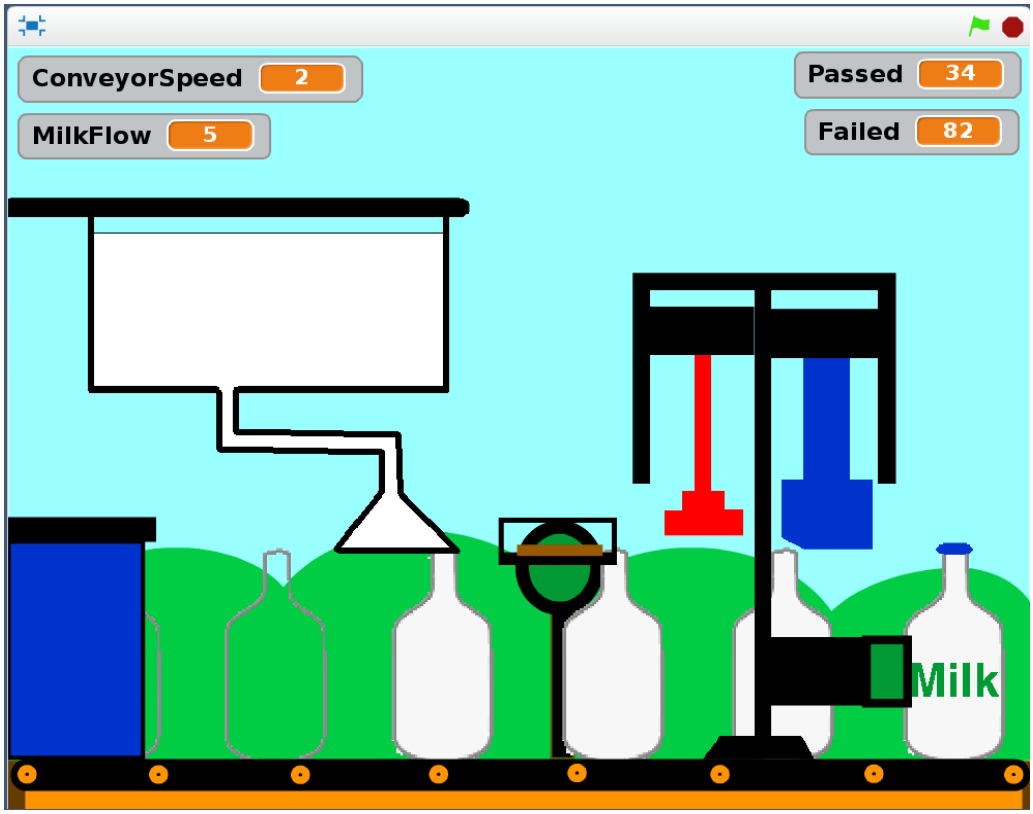
Longlevens Library Code Club

Scratch - Milk Bottle Factory

Description

This is a Scratch program that shows how a processing system operates. Empty bottles are fed along a conveyor belt, they fill up with milk as they pass under a filling machine and are then checked that they have been correctly filled up to the top. If they are full, they continue along the production line to be sealed with tops and labelled, otherwise they are removed for recycling. The conveyor belt speed and milk flow rate can be adjusted to find the most efficient speed to produce finished bottles.



Load and Save the Resources

Open Scratch and load the “MilkBottleFactoryResources.sb2” file. Then save the project under a new name in your own “Coders” folder. The resources project has all the sprites that we will be using for the placing the bottles on the conveyor belt, filling the bottles, checking the bottles are full, rejecting if not full or sealing and labelling them if they are full.

The Conveyor Belt

The speed of the conveyor belt determines how fast the milk bottles are move along the production line. The bottles will start on the left and progress to the right through the factory on the belt.

In the Conveyor Belt Sprite select the “Data” category and “Make a variable”. Call the variable “ConveyorSpeed” and leave the variable set “For all sprites”.

We need to control the speed, so go to the “Control” category and drag into the Scripts area the following elements:

“When <Green Flag> clicked”

“When <space> key pressed” - change the key to “<left arrow>”

“When <space> key pressed” - change the key to “<right arrow>”

Under “When <Green Flag> clicked” add from the “Looks” category; “go to front”. This ensures the sprite is not hidden behind any other sprites. Then using the “Data” category set the initial “ConveyorSpeed” to “5” underneath “go to front”.

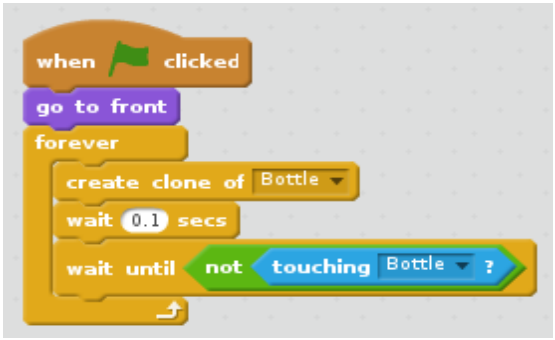
Also from the “Data” category, choose the “Change <ConveyorSpeed> by <value>” element and use it to change the “ConveyorSpeed” by “1” when the “<right arrow>” key is pressed and “-1” when the “<left arrow>” is pressed.

The Bottle Placer

The Bottle Placer’s function is to put a new empty bottle on the conveyor belt when the previous bottle has moved on the conveyor belt to a point clear of the Placer. Select the Placer sprite then from the “Control” category drag “When <Green Flag> clicked” into the scripts area. To make sure the illusion isn’t spoilt add a “go to front” elements from the “Looks” category so that bottles will appear behind it. Drag the “forever” loop element from the “Control” category and snap it under the “go to front” element. From the same category drag into the loop a “create clone of <sprite>” element and set the type of sprite to “Bottle”.

Before the Placer creates another bottle it needs to wait until the new bottle has left the Placer. Insert into the loop a “wait until <condition>” element from the “Control” category. From the “Operators” category, place a “not <condition>” into the “wait until” element. Then from the “Sensing” category, drag over a “touching <what>?” element into the “not” element.

We need to make sure the loop doesn’t run too fast causing other scripts to run slowly. Put a “wait <seconds>” element from the “Control” category into the loop between the “create clone of Bottle” and the “wait until” elements. Set the time to wait to “0.1” seconds. As the bottle clone takes time to create, without that pause the test for the latest bottle touching the Placer may return the wrong value and a second or even third bottle may appear on the conveyor belt before the first bottle leaves the Placer.



Creating and Moving the Bottles

The Bottle sprite is going to control how the bottles are processed. As the bottles move along the conveyor belt different actions will be triggered as the bottles come into contact with other sprites.

To keep control we need to know what is happening with each cloned bottle as it passes along the conveyor belt. For this we will use a data variable called “myBottleStatus”. We will keep updating this as the Bottle moves along the Conveyor Belt sprite coming into contact with the other sprites.

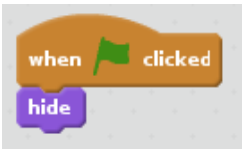
Cloned Bottle Initialisation

Select the Bottle sprite and create the following variables using the “Data” category:

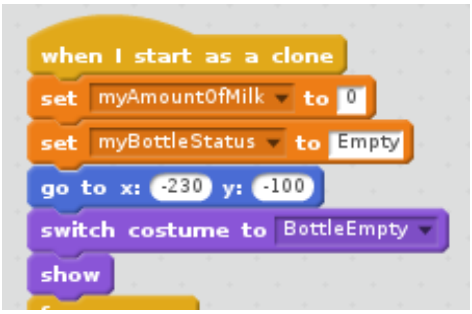
“myBottleStatus” (for **this sprite only**) - This indicates the stage the bottle has reached.

“myAmountOfMilk” (for **this sprite only**) - The quantity of milk in each bottle.

We only want to move the clones of the Bottle sprite created by the Placer sprite. The original Bottle sprite won’t be used and needs to be hidden from view. From the “Events” category drag a “when <Green Flag> clicked“ element to the script area. Then from the “Looks” category, drag a “hide” element and lock it below the “when <Green Flag> clicked”.



To initialise cloned bottles, drag over into the scripts tab a “when I start as a clone” element from the “Events” category. Set the “myAmountOfMilk” variable to zero and “myBottleStatus” to “Empty” using “set <variable> to <value>” elements.



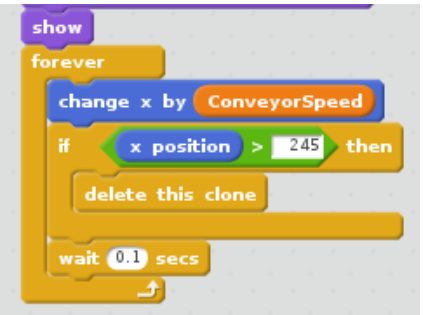
The bottle needs to start at the left hand side of the conveyor belt behind the Bottle Placer sprite. Use a “go to x: <value> y: <value>” element from the “Motion” category and set the value of x to “-230” and y to “-100”.

Switch the costume to “BottleEmpty” and add a “show” element using the “Looks” category.

Moving the Bottles

To move each bottle on the conveyor belt we need to add a forever loop that increases the “x position” by the speed of the belt. From the “Control” category lock a “forever” loop under the “show” element and lock into the loop a “change x by <value>.” element from the “Motion” category. Set the value of the element to be the conveyor belt speed by dragging the “ConveyorSpeed” data value from the “Data” category. To stop the loop running too fast, snap a “wait <value> secs” element from the “Control” category under the if statement and set the value “0.1” seconds.

When the bottle gets to the end of the conveyor belt we need to delete it. From the “Control” category, choose the “if <condition> then” element and drag that so it locks under the “change x by ConveyorSpeed” element. From the “Operators” category, choose the “ > ” operator and drag it into the <condition> part of the “if” element. In the left hand side of the greater than operator, drag and lock in the “x position” element from the “Motion” category and put “245” in the right hand side. From the “Control” category, drag “delete this clone” into the body of the “if” statement.



Now test your script and check that empty milk bottles appear behind the “Placer” sprite and disappear after they have reached the end of the conveyor belt.

The Filler

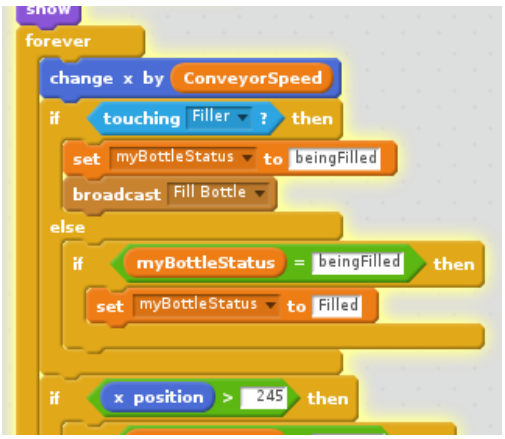
This is a large vat of milk that fills up each bottle as it passes underneath the Filler's nozzle. Select the Filler sprite, then from the "Data" category create the variable "MilkFlow" (for all sprites). This represents how fast the milk is flowing out of the filler.

Repeat the instructions for the Conveyor Belt but use the <up arrow> and <down arrow> keys to increase and decrease the flow of milk. Set the "MilkFlow" variable to "5" when the Green Flag is clicked. Add an "if" element from the "Control" category to prevent the value of MilkFlow being less than zero. You might want to go back and add a similar "if ConveyorSpeed > 0 then" element to the Conveyor Belt to stop it from going backwards.

Filling the Bottles

As each bottle passes under the Filler it will come into contact with the Filler's nozzle. The bottle touching the nozzle will trigger the filling of the bottle with milk. Milk flow needs to cease once the bottle is full or when it has been moved too far by the conveyor belt to remain in contact with the nozzle.

Go back to the Bottle sprite and insert an "if <condition> then <script1> else <script2>" element into the loop after the "change x by ConveyorSpeed" element. Insert into the condition part of the "if" element a "touching <what>?" element from the "Sensing" category. Set the value in this element to "Filler". Change the "myBottleStatus" to "beingFilled" as the only bottle we want to fill is the one touching the Filler.



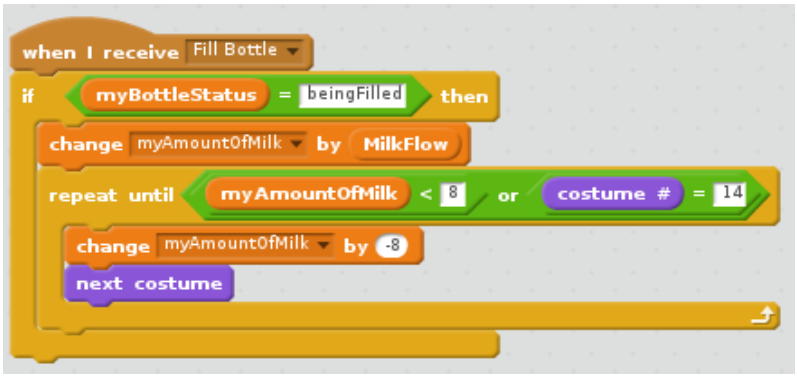
Filling needs to be performed separately from the motion loop, so the event of touching the Filler requires a message to be broadcast for activating the filling operation. From the "Events" category, drag "broadcast <message>" into "<script1>" of the "if touching Filler?" element. Set the message to be "Fill Bottle". Once the bottle is no longer in contact with the Filler, we need to move the status on from "beingFilled" to "Filled". Insert into the "else <script2>" part of the "if touching Filler?" element an "if <condition> then" element. Use the "if" element's condition statement to check that the variable "myBottleStatus" equals "beingFilled" and after the "then", set "myBottleStatus" to "Filled".

To fill the bottle, start a new script for the Bottle sprite with the "when I receive <message>" element from the "Events" category. Set the message to be "Fill Bottle". We need to check that the bottle is the one being filled. Add an "if <condition> then" element and check "myBottleStatus" equals "beingFilled. In the "if" element, add from the "Data" category a "change <myAmountOfMilk> by <MilkFlow>" element.

As the milk enters the bottle we need to animate the level of milk in the bottle rising. The Bottle sprite comes with 14 costumes in rising order of milk levels from empty to full. When the amount of milk in the bottle reaches "8", the next costume in the sequence shall be chosen. To know when the costume needs changing again, we deduct 8 from the "myAmountOfMilk" variable. We could use an "if" statement to check the value of the "myAmountOfMilk" variable, but if the "MilkFlow" is

greater than 8 we may need to change up two or more costumes at once. It is therefore better to use a “repeat until <condition>” loop from the “Control” category. The loop runs until “myAmountOfMilk” is less than 8 or when the bottle is full.

To check if the bottle is full, use the special “costume #” variable from the “Looks” category. This holds the current costume number, number 14 being the costume for a full bottle. Insert an “ or “ operator in the “repeat until” loop’s condition before adding the checks for “myAmountOfMilk” being less than 8 and “costume #” being “14”.



Before continuing, test your code by clicking the Green Flag and check the amount of milk in a bottle appears to rise as it passes under the Filler. What happens when you speed up or slow down the conveyor belt?

The Monitor

The Monitor sprite indicates if the bottle in front of it is full or not full of milk and counts them. Select the sprite and then from the “Data” category create the following variables:

“Passed” (for all sprites) - This will count how many bottles are correctly filled.

“Failed” (for all sprites) - This will count how many bottles are under filled.

“bottleChecked” (for this sprite only) – Indicates if the current bottle has been checked.

Set the “Passed” and “Failed” variables to zero and “bottleChecked” to “False” when the Green Flag is clicked. The “bottleChecked” variable is a special Boolean variable which we will use to stop a bottle from being checked and counted more than once.

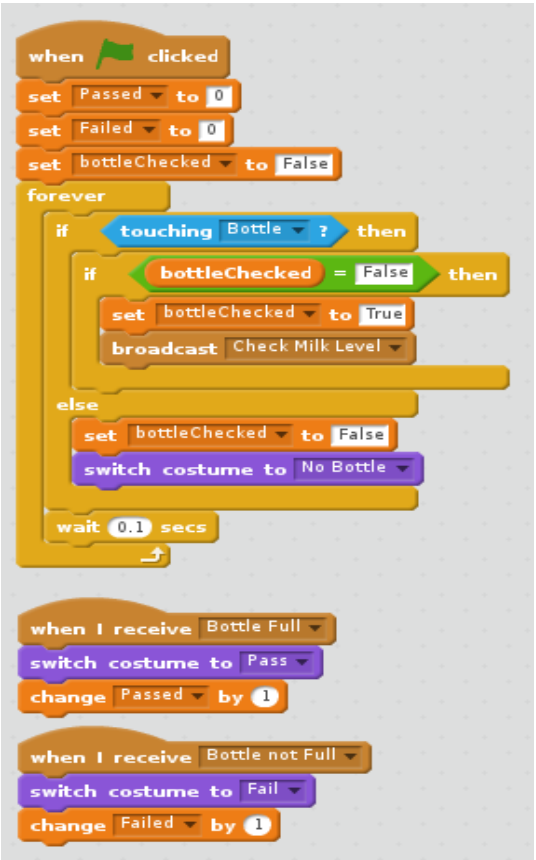
The Monitor has three different appearances:

1) No bottle touching – “No Bottle” costume

2) Full bottle – “Pass” costume

3) Partially full bottle – “Fail” costume

Add a “forever” loop from the “Control” category and add into it an “if <condition> then <script1> else <script2>” element. Insert into the <condition> a “touching <what>?” element from the “Sensing” category. Select “Bottle” as the item that is being sensed. This will continuously check if a bottle is in front of the Monitor. Insert into “<script1>” an “if <condition> then” element. Add an “ = ” operator from “Operators” into this “if” element’s <condition> and use it to check if “bottleChecked” is “False”. If it is we need

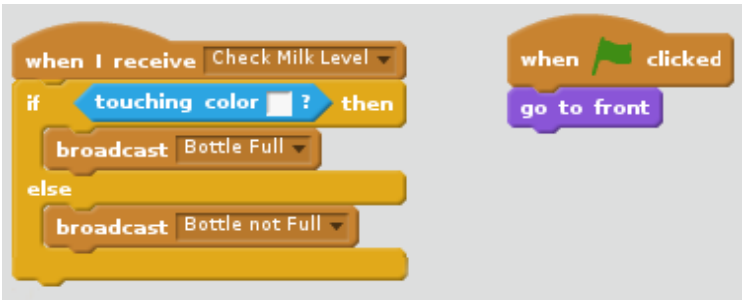


the “Sensor” sprite to check if there is milk in the bottle right up to the top. Change the “bottleChecked” variable to “True” and then from “Events”, drag over a “broadcast <message>” element and set the message to “Check Milk Level”.

If no bottles are touching the Monitor it shall return to its “No Bottle” costume. In the “else <script2>” part for the condition “touching Bottle?” reset the “bottleChecked” variable to “False” and then insert a “switch costume to <No Bottle>” from the “Looks” category. To reduce the number of times the loop runs per second, add a “wait 0.1 secs” element.

When the Sensor sprite checks if there is milk at the top of a bottle it shall broadcast either the message “Bottle Full” or “Bottle not Full”. From the “Events” category, drag over the “When I receive <message>” element twice. In the first one select the message “Bottle Full”. In the second one select the message “Bottle not Full”. Under each add a “switch costume to <costume name>” element. For “Bottle Full” set the costume to “Pass” and add underneath a “change <Passed> by 1” element from the “Data” Category. Repeat for “Bottle not Full” but the costume is “Fail” and change the “Failed” variable by 1.

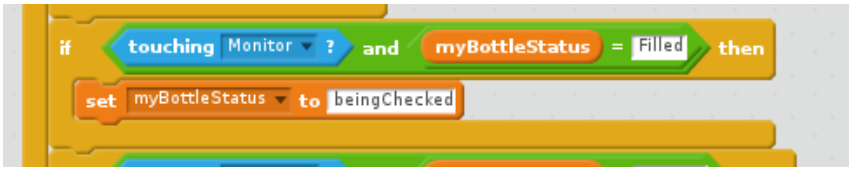
The Sensor



The Sensor sprite determines if there is milk right up to the top of the bottle currently passing between it and the Monitor sprite. We set up the Monitor to broadcast a “Check Milk Level” message whenever there is a bottle in front of it. We therefore need to trigger a script starting with “when I receive <Check Milk Level>” element from “Events”. Under it add an “if <condition> then <script1> else <script2>” element. The condition is whether the sensor is touching the white colour of the milk in the bottle so insert in a “touching color?” element and change the colour to the same colour as used for milk in the bottle costumes. In <script1> insert a “broadcast <Bottle Full”> element. In <script2> insert a “broadcast not Full”.

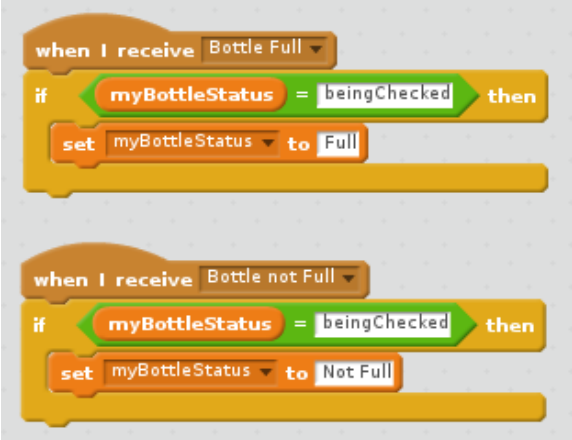
We want the Sensor to be positioned in front of the bottles so add in a “when <green flag> clicked” element with “got to front” snapped under it.

Checking the Bottles



When a bottle passes the Monitor we trigger the checking by broadcasting a “Check Milk Level” message. In the Bottle sprite, we need to set the status of the bottle to “beingChecked” to make sure only that bottle is checked. Add into the main “Forever” loop an “if <condition> then” and insert in an “and” operator. Set the left hand condition of the “and” operator to “touching <Monitor>?” and in the right add a check for “myBottleStatus” equals “Filled”. In the body of the “if” element set the “myBottleStatus” to be “beingChecked”.

The Sensor broadcasts either “Bottle Full” or “Bottle not Full” which we used in the Monitor scripts but we also need to use them in the Bottle scripts. Use a “when I receive <message>” element from “Events” and set the message “Bottle Full”. Add an “if <condition> then” element. Set the condition to the “ = ” operator and use it to check that the bottle has “myBottleStatus” equal to “beingChecked”. If it is, set the “myBottleStatus” to “Full”. Duplicate the script but change the message to “Bottle not Full” and set the new “myBottleStatus” to “Not Full”.



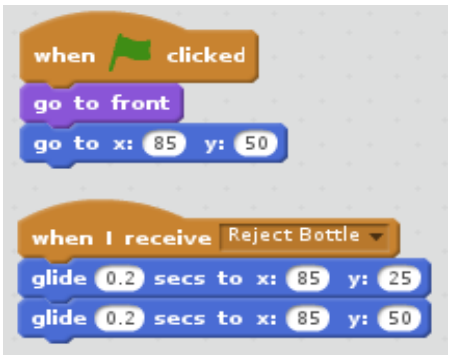
Click on the Green Flag and check that when full bottles pass the Monitor, it turns green and the number of “Passed” bottles increments by 1. Speed up the conveyor belt so that bottles fail to become full with Milk. Check that as they pass the Monitor, the Monitor turns red and the number of “Failed” bottles increments by 1.

The Finisher

After a bottle has been checked it needs to be removed if it is not full, otherwise it carries on to be labelled and sealed. The Finisher sprite is simply a framework for three other sprites, the “Knocker”, the “Topper” and the “Printer”. The only purpose of the Finisher sprite itself is to be used to sense when a bottle is touching it. There is no code to write on the scripts tab for this Sprite.

The Knocker

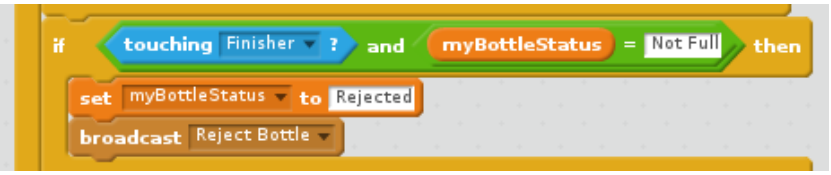
The knocker will knock off any bottle that is not full from the conveyor belt. It activates when a bottle that is not full is touching the Finisher. It needs to be in the up position when the Green Flag is clicked and in front of the bottles. Choose the “When <green flag> is clicked” element and snap underneath a “go to front” element followed by a “go to x: <value> y: <value>” element from the “Motion” category. Set the “x” value to be 85 and the “y” value to be 50 so that it fits snugly into the front part of the finisher.



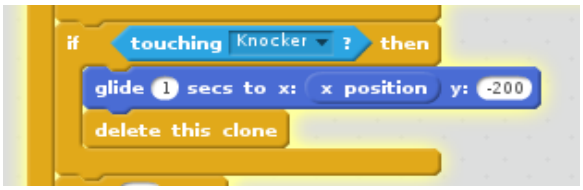
The Knocker will be summoned into action by the message “Reject Bottle” being broadcast. Select “When I receive <message>” element from the “Events” category and add underneath two “Glide <0.2> seconds to x <value> y <value>” elements from the “Motion” category. Set the “x” value to “85” in both and then set the “y” value to “25” in the first one and “50” in the second.

Rejecting Bottles

The action of the Knocker is triggered when a bottle touches the Finisher and has “myBottleStatus” set to “Not full”. The message “Reject Bottle” shall be broadcast from the Bottle sprite. Add the touching and status checks in a new “if <condition> then” element in the Bottle’s main “forever” loop. Add into the “if then” element’s main body, a change of “myBottleStatus” to “Rejected” and broadcast of the message “Reject Bottle”.

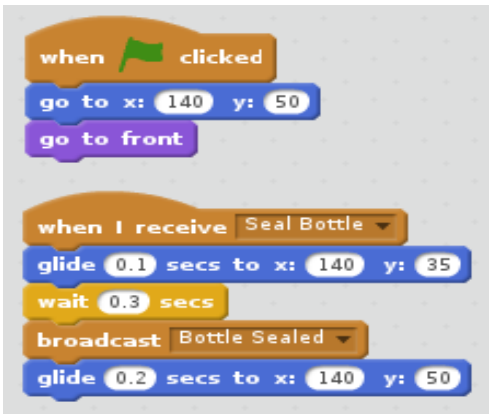


When the Knocker receives “Reject Bottle” it moves downwards coming into contact with the Bottle underneath. The Bottle sprite needs to react when the Knocker touches it. Add yet another “if <condition> then” element to the Bottle sprite’s forever loop and set the condition to “touching Knocker?”. Add a “glide <1> secs to x:<value> y: <-200>” element into the main body of the “if” element. We won’t complicate the “x:” value, we’ll just keep it the same as it was when the Knocker makes contact. Drag over from “Motion” the special “x position” variable. After it has moved down behind the conveyor belt we simply need to delete it, so add a “delete this clone” element under the “glide” element.



The Topper

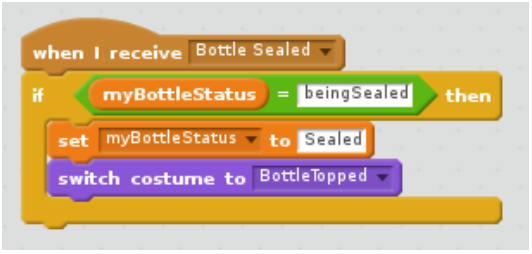
The Topper will add a top to seal the bottles as they pass underneath. It moves in the same way as the knocker. Copy the code from the Knocker and change all the x values to “140”, the y value that is “25” to “35” and the message in the “when I receive <message>” to “Seal Bottle”. When the Topper is in the down position, add in a “broadcast <Bottle Sealed> element. This will trigger the bottle to change to the “Bottle Sealed” costume. Change the glide time to “0.1” seconds for the initial “glide” element as we want the pressing down action to be quick. Also add a “wait <0.3> secs” element so that the seal gets a firm press!



Sealing the Bottles

When a bottle passes the Topper sprite the top of the bottle shall just touch it triggering the broadcast to the Topper sprite to seal the bottle. The bottle’s “myBottleStatus” must be “Full” and immediately set to “beingSealed” to stop repeat broadcasts.

When the “Topper” broadcasts “Bottle Sealed” we want to swap the costume to a new sealed costume. Select the full bottle costume with the mouse and then click the right hand button on the mouse. Choose “duplicate” and on the new costume (number 15) draw a top on the bottle and rename the costume to a suitable new name. Start a new script in the Bottle sprite with “when I receive “Bottle Sealed”, check the status of the bottle is “being Sealed” and then set the status to “Sealed” and switch the costume to the new costume.



Click the Green Flag and check that full bottles become sealed as they pass the Topper. Check what happens if you temporarily remove the if statement to check the status is “beingSealed”. Do strange things start happening to the other bottles? This is why we need to make sure the status of each bottle is tracked so that each bottle knows what stage of processing they have passed and which stage is appropriate next. Make sure you restore the blocks before you continue.

The Printer

The Printer adds the label “Milk” to the bottles as they pass. The operation is triggered when a bottle touches the Printer and broadcasts the message “Print Label”.

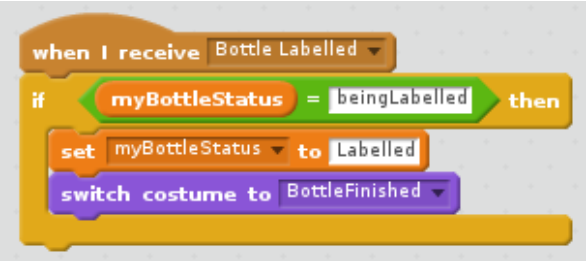
Use a “When <Green Flag> clicked” script to bring the Printer to the front and initialise the costume to “Off”.

Then drag over from “Events” a “when I receive <message>” element and change the message to “Print Label”. Switch the Printer’s costume to “On”, broadcast a message “Bottle Labelled” then insert a “wait until <condition> element”. We are going to turn the Printer back to the “Off” costume when the bottle has passed the Printer. Insert a “not <condition> and into that insert a “touching <Printer>?” element. After that add on a “switch costume to <Off>” element.

Labelling the Bottles

To trigger the Printer, inset another “if” statement into the “forever” loop and use it to check if the bottle is touching the Printer and has the status “Sealed”. Change the status to “beingLabelled” and broadcast “Print Label” so that the Printer reacts.

In the costumes, duplicate the bottle with the top and add the word “Milk” in roughly the centre of the bottle and name it “BottleFinished”. When the Printer sends back the message “Bottle Labelled” we want to switch to this new costume and set the status of the bottle to “Labelled”. The switch takes place behind the bracket holding the Printer so the transformation from unlabelled to labelled is hidden from view and the bottle appears with its label as it moves out from behind the Printer.



Finding the best conveyor belt speed and milk flow

The bottle’s production line is now complete. Click the Green Flag and increase the conveyor belt speed and the milk flow rate and see if the sprites can keep up with all the actions necessary to allow full, sealed and labelled bottles to reach the end of the conveyor belt. At some point an unfilled bottle will reach the end of the conveyor belt which would be a very messy situation. Insert an action within the main “forever” loop to deal with the problem by shutting down the processing.

Additional Challenges

A. Convert the factory into a game

Add a timer to make the factory into a game to see who can achieve the most full bottles. Maybe incur a time penalty when bottles are knocked off the conveyor belt for not being full. The game ought to end if an unfinished bottle gets to the end of the conveyor belt due to running the conveyor belt too fast, but you could add a reset button and a time penalty.

B. Improve the animations

1. Make the quantity of milk in the Filler's vat decrease as bottles are filled. Refill it from a tanker lorry when it gets too low. Decrease the maximum value of "MilkFlow" when the level of milk in the vat is running low.

2. The current nozzle would spill lots of milk. Make a nozzle that moves with the bottle or stop the bottle by lifting it off the conveyor belt temporarily while filling it, fitting it snugly into the nozzle to stop leaks. Make sure if you stop the bottle to fill it, the next bottle on the conveyor belt doesn't hit it.

3. Rather than the rejected bottles falling off the conveyor belt, add another conveyor belt to take them back to the beginning to be filled up properly on a second attempt.

4. Make the bottle Placer indicate when a new bottle is about to appear and add a counter for how many bottles it has placed since the Green Flag was clicked.

C. Create a different factory

Can you apply what you learnt in the Milk Factory to another type of factory? Perhaps a cream cake or biscuit factory or a car assembly plant where you add the major items such as the wheels and doors.