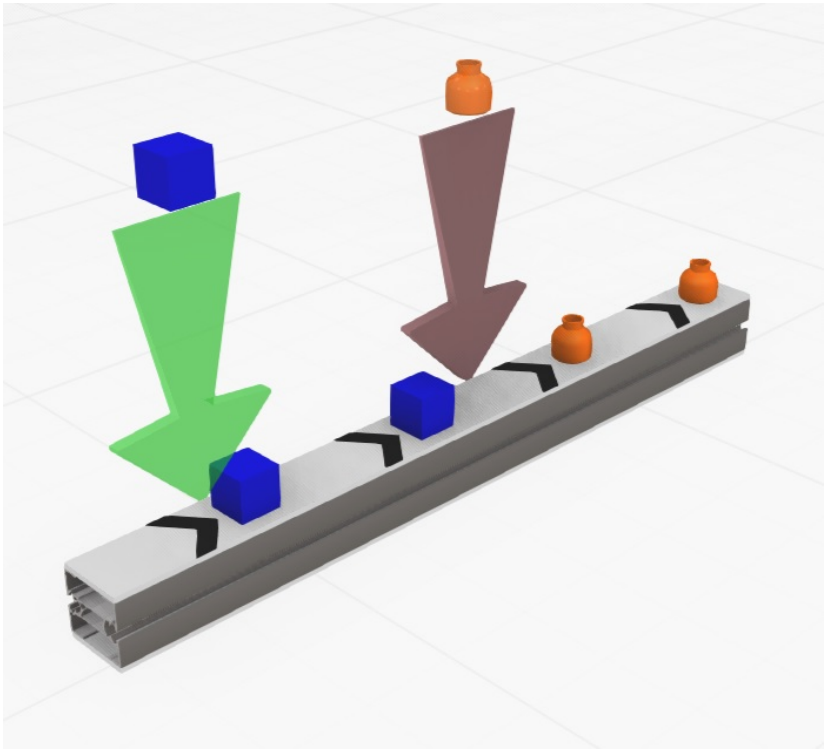
Simulation components (Feeder, Changer & InitializationPoint)

Last Modified on 06/11/2019 8:47 am CET

This article describes the simulation components Feeder, Changer and InitializationPoint focusing on some simple examples. The User documentation (manual) is also covering this area. These three components can be used to get a wide range of behaviours on a system simulation, both in terms of visual appearance of the products but also how the products are routed in a system.

Example 1: Replace (One-to-one)

The first example is a really simple case showing how products can change appearance at a certain point.

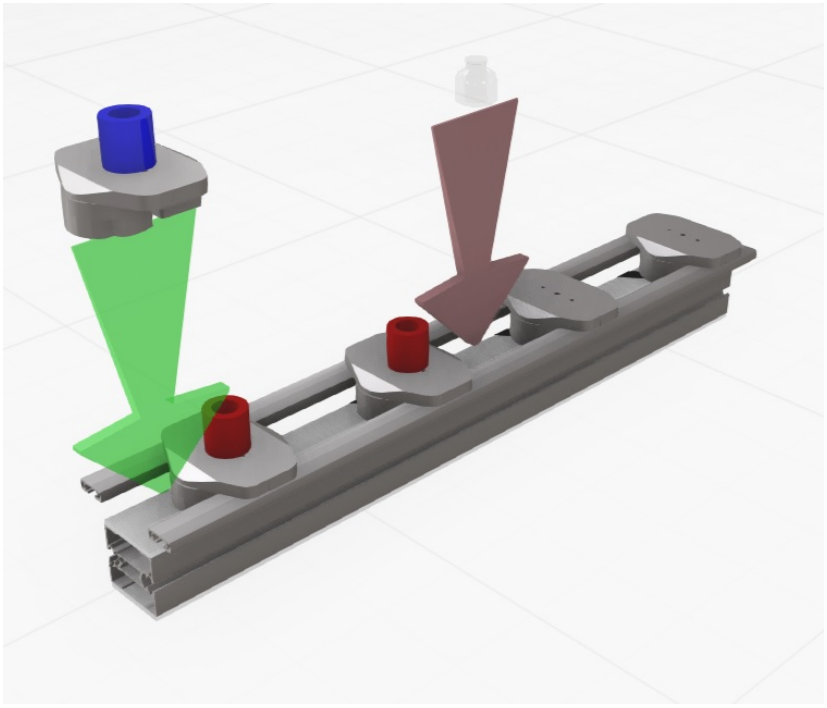


The ***Feeder*** component is set to create products in the shape of a box. The ***InitList*** is not used which results in products in the default color (Blue).

The ***Changer*** component is set to ***Change: All*** components. The ***Behaviour: OneToOne*** means there is exactly the same amount of products before and after the component. The ***Mode: Replace*** is set which will remove the incoming products and add the new product defined in the Changer component.

Example 2: Remove (One-to-one)

The second example is a bit more features. Products are removed at the point of the changer.



The Feeder component is now using the InitList. This list is describing the destination route of each product. The syntax is a separate row for each product route in the following format:

”nr of products”, ”color”, ”destination1”, “destination2”, …

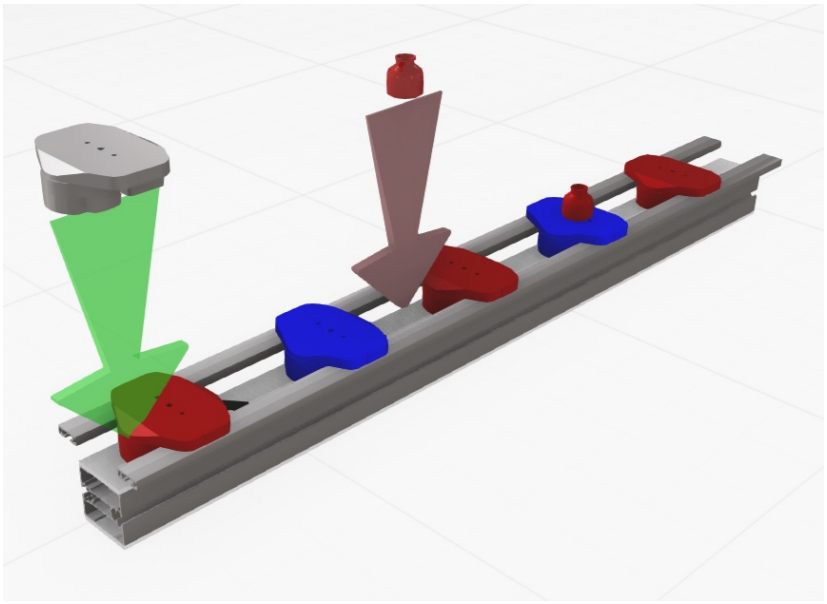
The Feeder component will create all products of the first row and the read the second line. After all lines are executed all lines are repeated continuously. The number of destination can be arbitrary. In this example the destination is not used but in order to get products in a specific color the Feeder has to be using the InitList. It will repeatedly create one red product with no specific destination.

1,red,

Then the Changer component is set to ***Mode: Remove***. Still this is unconditional so it will remove all products. The platform is also changed to a X85P which makes it more visible that the products are removed from the pallets. There is also a setting to remove also the pallet if that is required.

Example 3: Add (One-to-one)

Next example is showing the how to add products to a pallet or puck.



This is done using the ***Mode: Add*** feature. This mode is only meaningful in pallet/puck systems. It adds a product to an empty pallet/puck. It is in this example combined with a condition ***Change: BasedOnID*** and the ***ProcessID*** is set to process2. The InitList on the Feeder component is used for creating products with two different destinations each with a separate color:

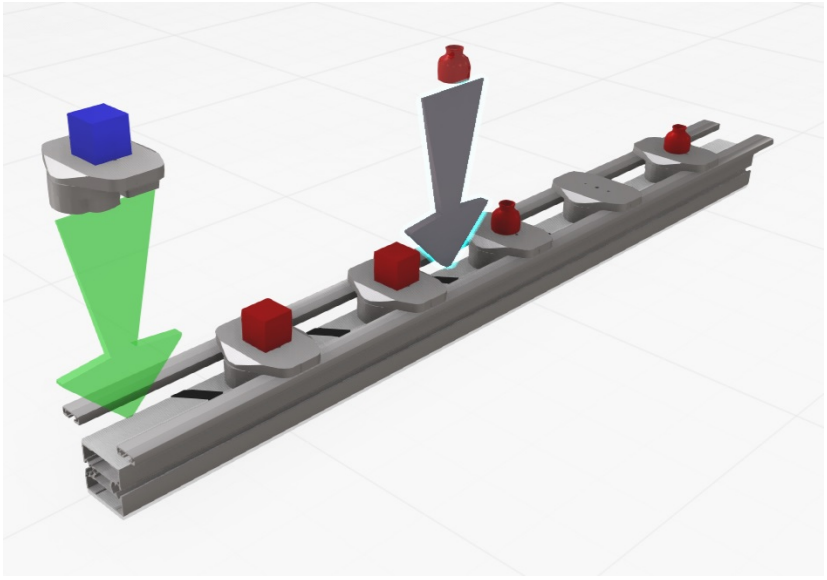
1,red,process1

1,blue,process2

Also the option ***IncludeProduct*** is deselected. Because there is no product to paint with the color (indicating the destination route) the whole pallet/puck gets the specified color. The result of this setup is that the changer will add a product (bottle) to all blue pallets (which is heading for process2).

Exemple 4: Batching

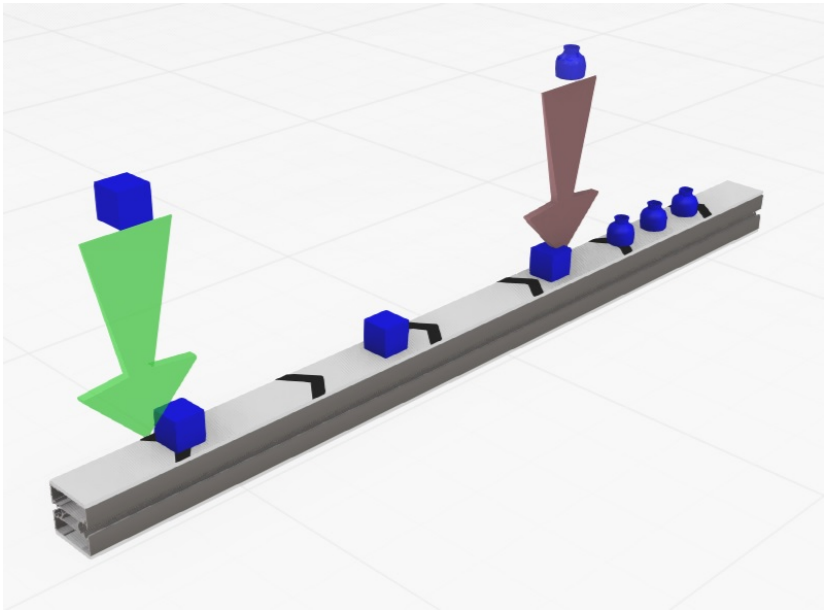
The batching function works on both pallet/pucks and ordinary conveyors.



To show batching the Feeder is set back to only create red boxes without any specific destination. The Changer is set to ***Behaviour: Batching*** and no condition on the ID is used. To specify how many products a batch is containing the ***BatchProducts*** is set to 2. For Pallet/pucks systems it is possible to set if the empty pallet/puck should be removed (***RemovePallet***).

Example 5: Splitter

The splitter function works only on non pallet/pucks conveyors.



The splitter function works only on non pallet/pucks conveyors. The ***AmountOfProducts*** specifies how many products is going to be removed in order to add one new product. In this case it is set to 3. The ***ProcessTime*** is set to 0.500 s which defines the time between the individual products after the split.

Additional info about routing.

The products (mainly pallet/pucks with products) can have a predefined route. This is defined as a list of destinations in the InitList. This list can be defined in the Feeder component but there is also a separate ***InitializationPoint*** component doing exactly this. Even the Changer component can update the routes if the ***ReInit*** option is set. Every individual products has a ***Pointer*** referring to the current step in the list of destinations. The parameter ***ProdID*** is the next destination to be reached. After reaching a process point the pointer is increased by one and the ProdID is updated according to the predefined route. After reaching the last destination the InitList is starting over.

The InitializationPoint component adds an InitList to all products that doesn't have one. It has also a ProcessID so products can have the InitializationPoints as a destination step in the route. Then the route will be overwritten with the new route.