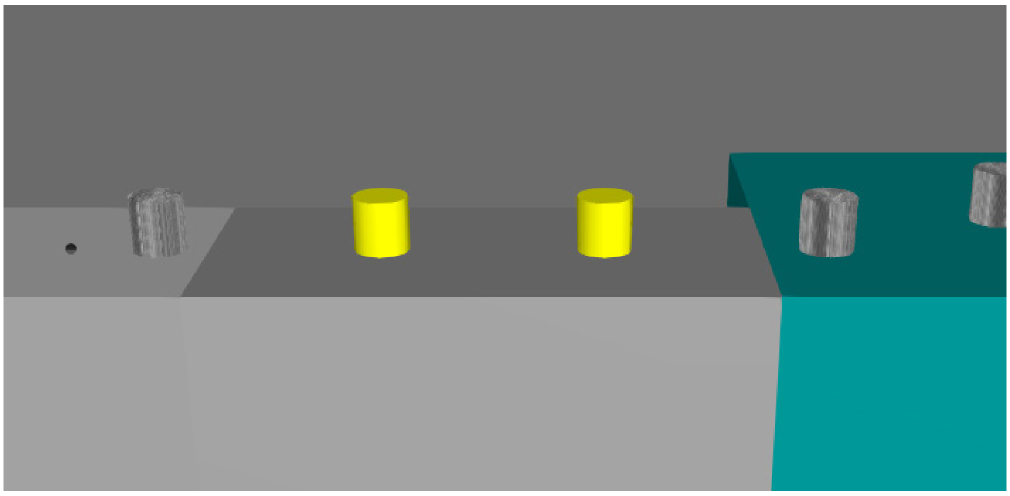
Highlight Components

Visual Components 4.0 | Version: February 28, 2017



During a simulation you may want to highlight components to provide important visual indicators. For example, you may want to highlight the stage of a component in a product life cycle or color code components in a workcell. It is also possible to highlight the state of a component, for example whether a machine or resource is active, idle or broken.

In this tutorial you learn how to:

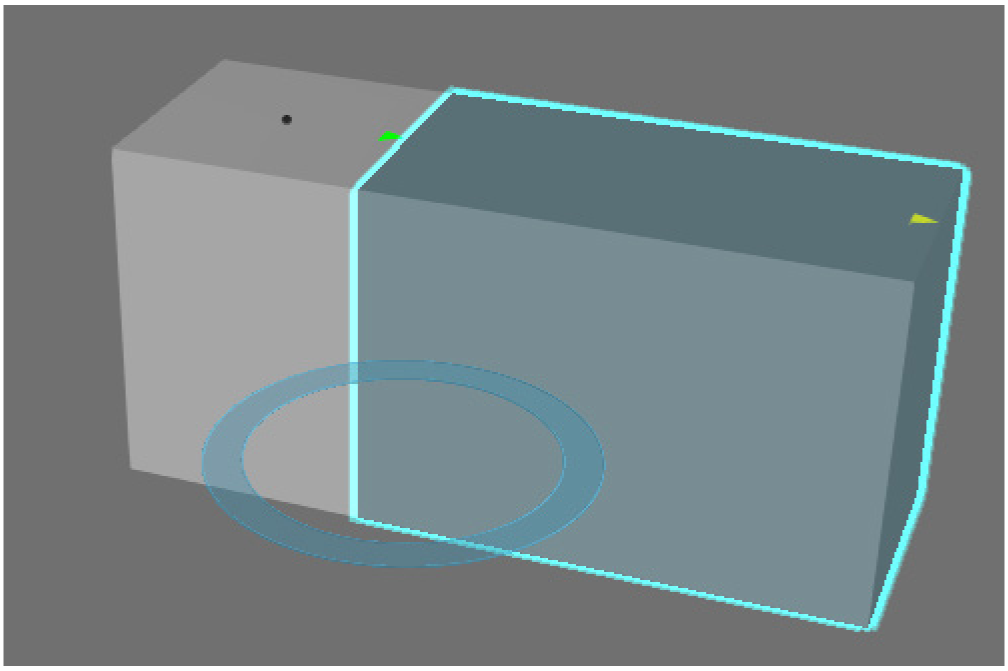
- Assign materials to components contained by paths and other containers.
- Use signals and events to edit components transitioning in and out of containers.
- Use a dictionary to store the property values of components.

Getting Started

1. Clear the layout of the 3D world.

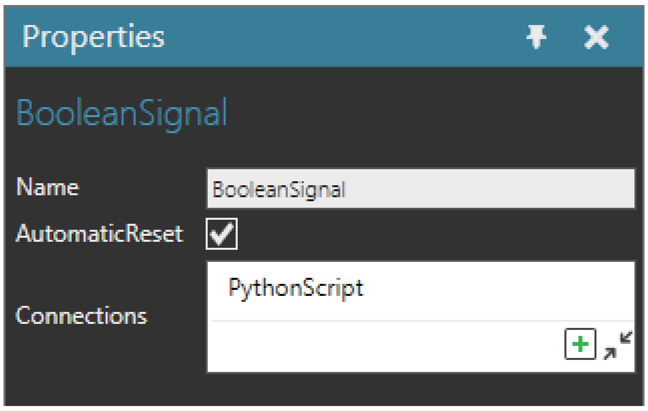
2. In the eCatalog panel, Collections view, under Models by Type, click **Component Templates**.

3. Add a **Feeder Template** to the 3D world, and then add and connect a **Conveyor Template** to the feeder in the 3D world.



4. With the conveyor selected in the 3D world, click the **Modeling** tab, and then add a **Python Script** behavior.

5. Add a **Boolean Signal** behavior, and then connect the signal to the **Python Script**, so the signal can be a trigger in the script.

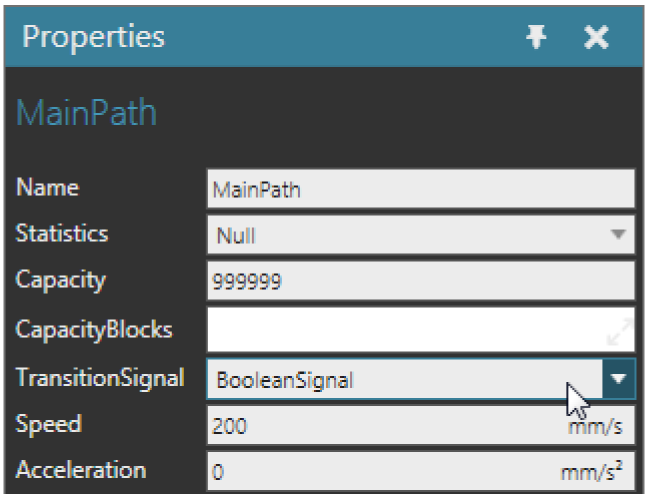


Properties

BooleanSignal

Name	BooleanSignal
AutomaticReset	✓
Connections	PythonScript

6. In the Component Graph panel, select the **MainPath** behavior, and then in the Properties panel, set TransitionSignal to the signal you created in step 5.



Name	MainPath
Statistics	Null
Capacity	999999
CapacityBlocks	
TransitionSignal	BooleanSignal
Speed	200 mm/s
Acceleration	0 mm/s²

A path with a transition signal allows you to know when a part enters and leaves the path through its ports.

Highlight First In, First Out

It is possible to highlight the first component that enters a path.

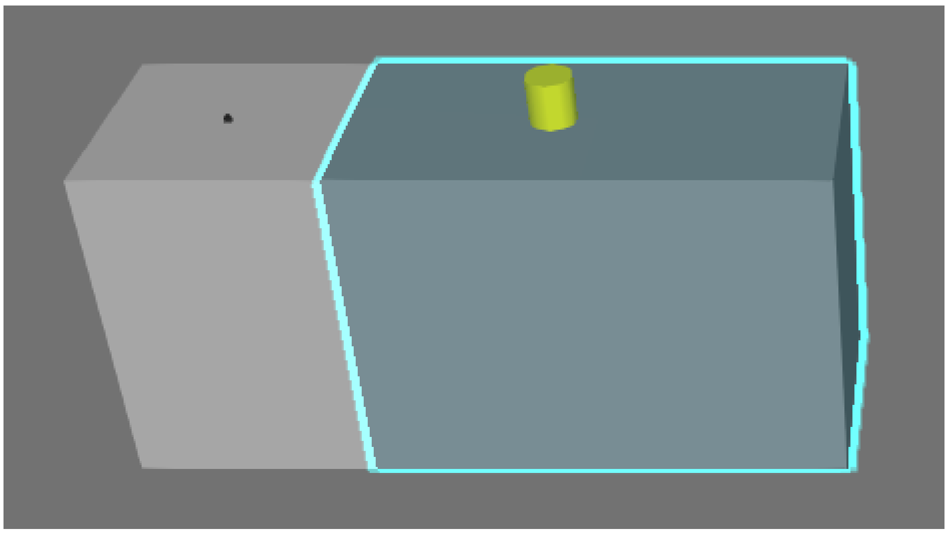
1. In the script editor, define variables for the component, main application, a yellow material, path and its transition signal.

```
from vcScript import *


app = getApplication()

yellow = app.findMaterial("yellow")

comp = getComponent()

path = comp.findBehaviour("MainPath")

transition_signal = path.TransitionSignal
```

2. In the OnRun event, create a while loop that changes the material of the first part that enters the path, and then compile the code.

```
def OnRun():
  while True:
    if transition_signal.Value == True:
      part = path.getComponent(0)
      part.NodeMaterial = yellow
      part.MaterialInheritance = VC_MATERIAL_FORCE_INHERIT
      delay(0.1)
    else:
      delay(0.1)
```

3. Run the simulation, verify the first part that enters the path is highlighted yellow, and then reset the simulation.



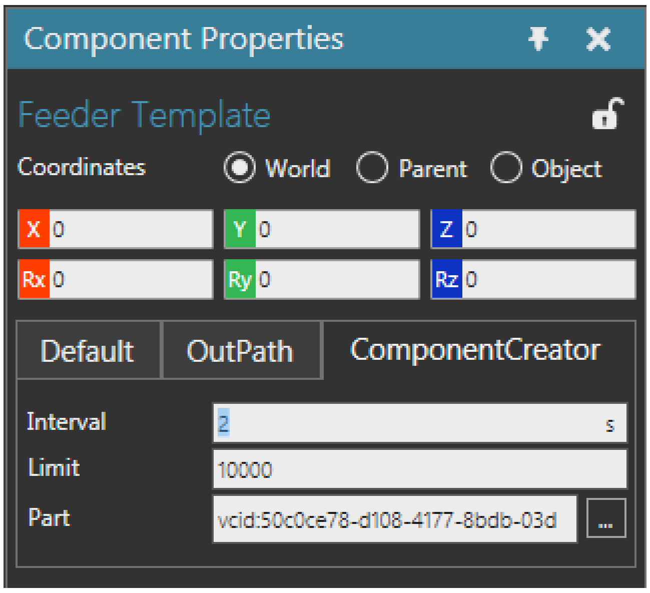
Highlight All

It is possible to highlight all components contained by a path.

1. In the 3D world, select the **feeder**.

NOTE! You do not need to close the script editor of the conveyor when modeling a different component.

2. In the Component Graph panel, select the **root node**, and then in the Component Properties panel, click the **ComponentCreator** tab, and then set Interval to **2**.



Component Properties

Feeder Template

Coordinates [✓ World] Parent Object

X 0 Y 0 Z 0

Rx 0 Ry 0 Rz 0

Interval 2 s

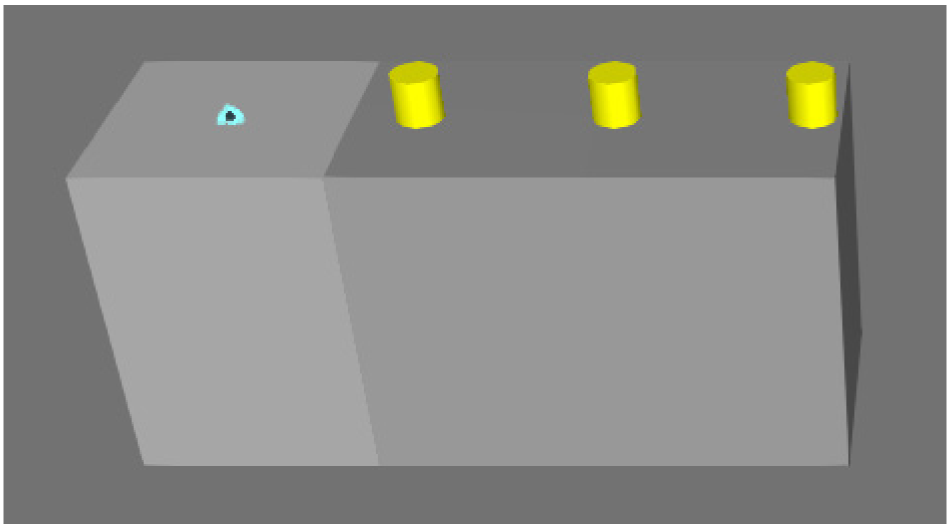
Limit 10000

Part vcid:50c0ce78-d108-4177-8bdb-03d

3. In the script editor, OnRun event, modify the code to change the material of all components contained by the path when its transition signal value is True, and then compile the code.

```
def OnRun():
 while True:
  if transition_signal.Value == True:
   for part in path.Components:
    part.NodeMaterial = yellow
    part.MaterialInheritance = VC_MATERIAL_FORCE_INHERIT
    delay(0.1)
  else:
   delay(0.1)
```

4. Run the simulation, verify all components that enter the path are highlighted yellow, and then reset the simulation.



This approach works, but you need to be careful when iterating through a list of components contained by a path. For example, parts may leave the path and in a high volume conveyor your code might degrade simulation performance. An alternative is to use the OnTransition event inherited by a path from vcContainer to highlight arriving parts.

5. In the script editor, create an event handler for the OnTransition event of the path which changes the material of an arriving part.

```
def highlightPartEnteringPath(part,isPartArriving):
  if isPartArriving == True:
    part.NodeMaterial = yellow
    part.MaterialInheritance = VC_MATERIAL_FORCE_INHERIT
...
path.OnTransition = highlightPartEnteringPath
```

6. In the OnRun event, clear the code and add a pass statement, and then compile the code.

```
def OnRun():
  pass
```

7. Run the simulation, verify the all components that enter the path are highlighted yellow, and then reset the simulation.

Undo Highlight

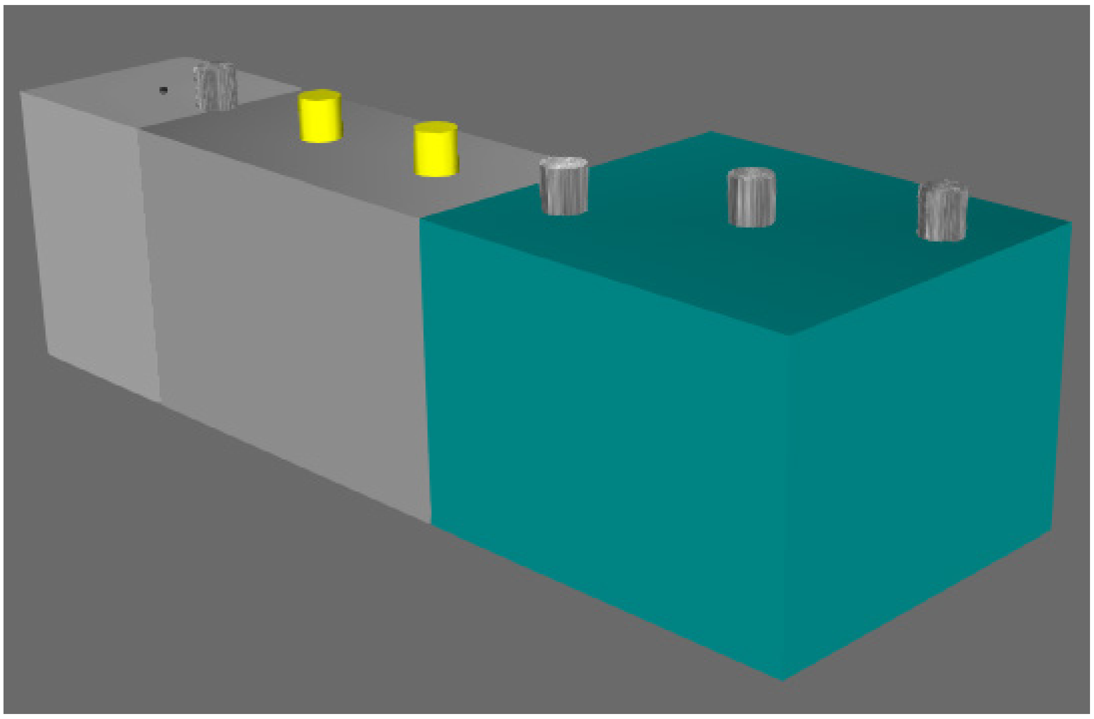
There are several ways to reset parts contained by a path to their original material. One way is to use the OnPhysicalTransition event of a vcMotionPath object.

1. In the script editor, create an event handler for the OnPhysicalTransition event of the path, and then compile the code. The event handler should get the material of a part entering the path and assign that same material back to the part when it leaves the path. In this case, it may be helpful to use a dictionary object to store the original material of a part.

```
original_materials = {}
def undoHighlightPartLeavingPath(path, part, isPartArriving):
  if isPartArriving == True:
    original_materials[part] = part.NodeMaterial
  elif isPartArriving == False:
    part.NodeMaterial = original_materials.get(part)
    part.MaterialInheritance = VC_MATERIAL_FORCE_INHERIT
...
path.OnPhysicalTransition = undoHighlightPartLeavingPath
```

2. Click the **Home** tab, and then add and connect an **Offset Conveyor Template** to the end of the conveyor in the 3D world.

3. Run the simulation, verify highlighted parts are reset to their original material, and then reset the simulation.



Review

In this tutorial you learned how to manipulate the materials of components moving along a path. You know how to use signals and events to identify parts entering and exiting paths and other containers. You also know how to create event handlers for recording and resetting the state of a component before and after it leaves a container.