Routing Rule Explanation_example

01. Here I have created an example workbench of car production line to explain some features of routing rule.

02. A feeder creates cars of 3 different colors. Red, Blue and White.

03. There is a behavior [Note] named (Stamping) in the feeder. The user can give inputs in the Note and then use the values as logic to generate dynamic component of his choice.
    Here the python script will read each character from the input string inside the [Note]-Behaviour and the use it to assign material to the created dynamic components (cars in this case) ---
    W = assign white color
    B = assign blue color
    R = assign red color

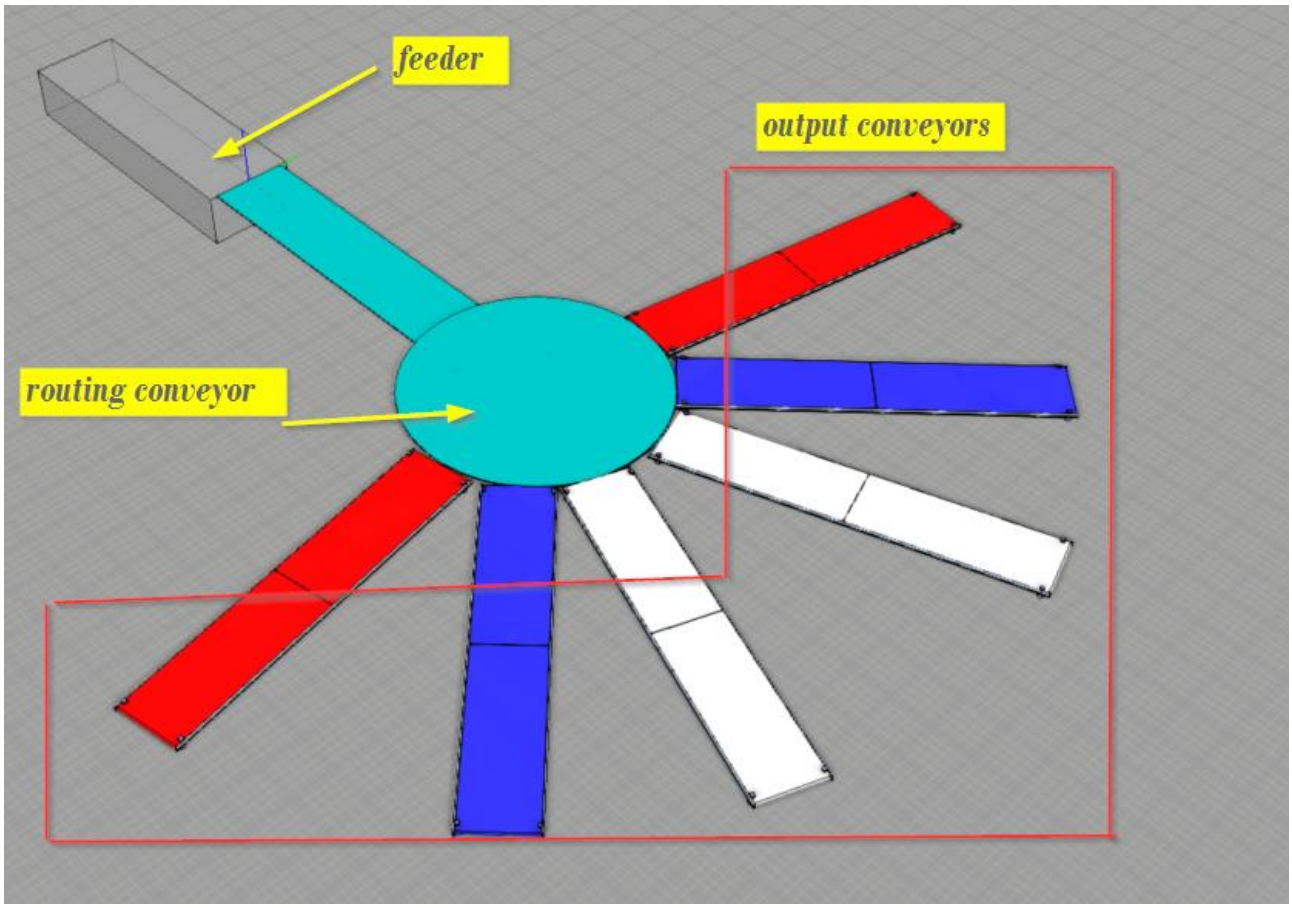
04. With every created dynamic component there is assigned 2 different properties into it. this means each car created from the feeder has 2 properties. Those are [CarID] and [Color]

05. Here is the code which generates those dynamic components (car's) with stamped properties –

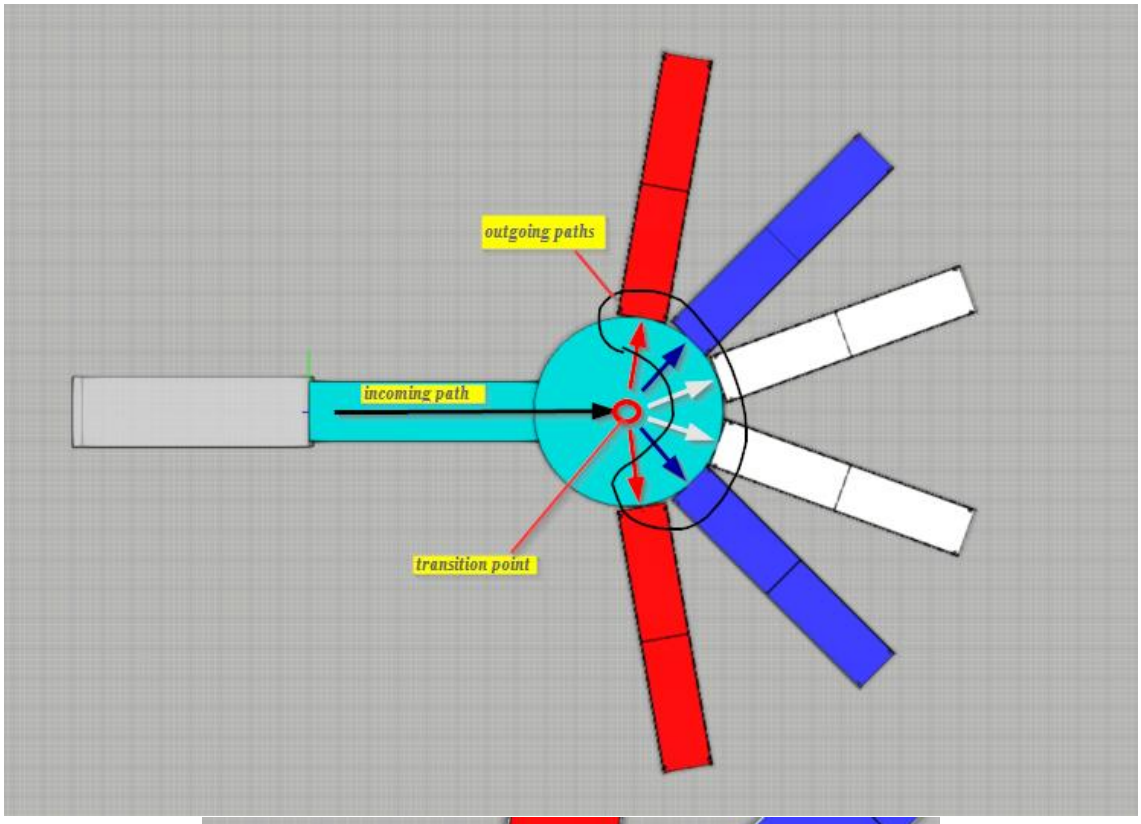
```
from vcScript import *

def OnSignal( signal ):
  pass

def OnRun():
  j = 101
  for i in stamping:
    print i
    creator.create()
    triggerCondition(lambda:getTrigger() == compSignal and compSignal.Value != None)
    car = compSignal.Value

    if i == "W":
      car.MaterialInheritance = VC_MATERIAL_FORCE_INHERIT
      car.NodeMaterial = white
      ###
      prop1 = car.createProperty(VC_STRING,'CarID')
      prop1.Value = 'white#%s'%(j)
      ###
      prop1 = car.createProperty(VC_STRING,'Color')
      prop1.Value = 'white'

    if i == "B":
      car.MaterialInheritance = VC_MATERIAL_FORCE_INHERIT
      car.NodeMaterial = blue
      ###
      prop1 = car.createProperty(VC_STRING,'CarID')
      prop1.Value = 'blue#%s'%(j)
      ###
      prop1 = car.createProperty(VC_STRING,'Color')
      prop1.Value = 'blue'

    if i == "R":
      car.MaterialInheritance = VC_MATERIAL_FORCE_INHERIT
      car.NodeMaterial = red
      ###
      prop1 = car.createProperty(VC_STRING,'CarID')
)p1.Value = 'red#%s'%(j)
      ###
      prop1 = car.createProperty(VC_STRING,'Color')
      prop1.Value = 'red'

    delay(delayTime)
    j+=1
  pass
```

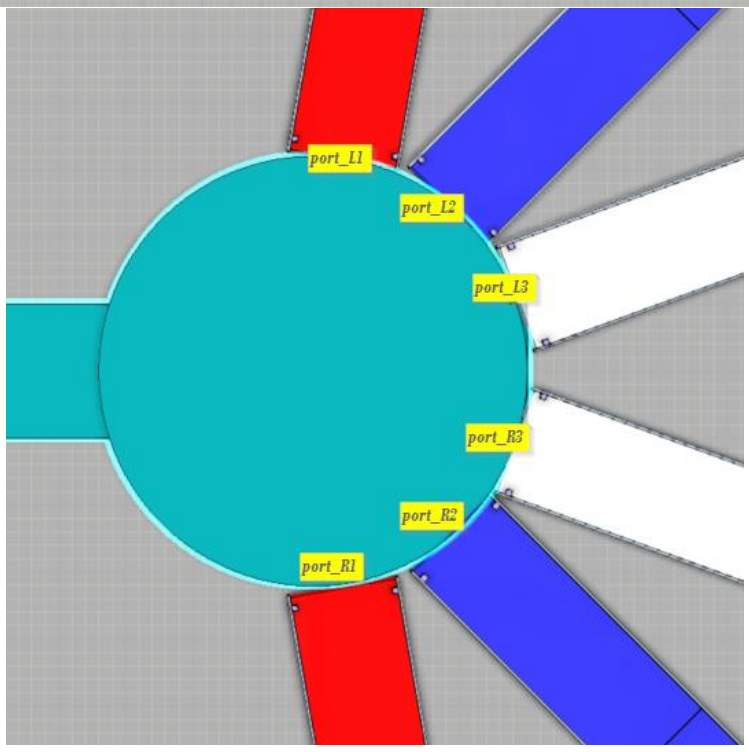
```
comp = getComponent()
app = getApplication()

white = app.findMaterial('white')
blue = app.findMaterial('blue')
red = app.findMaterial('red')

stamping = comp.getBehaviour('Stamping').Note
creator = comp.getBehaviour('Creator')
delayTime = creator.getProperty('Interval').Value
compSignal = comp.getBehaviour('ComponentSignal')
```

06. Then another customized component [routing_conveyor.vcm] was created. This conveyor has 1 incoming path and 6 outgoing paths. Two outgoing path is assigned for each color. Below is the screenshot.



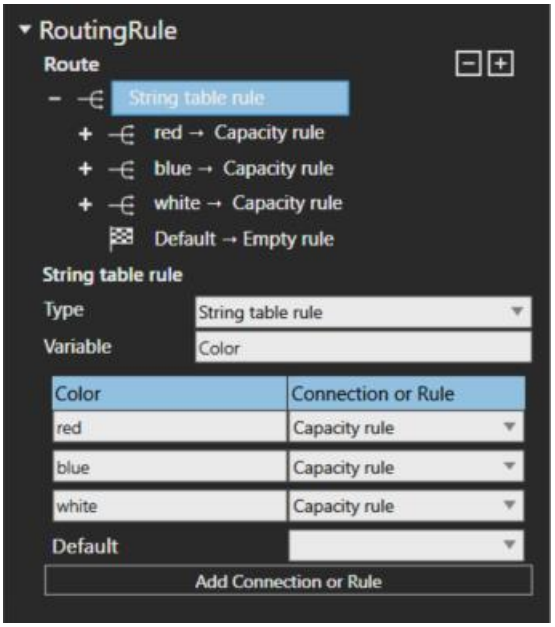
07. The routing rule is quiet simple to be configured, just the designer has to think one step at a time and then assigning condition one after another. In this conveyor first it checks the property [Color] of the component (car) moving on the path. Then it checks for the corresponding port's for the specific Color. That means it checks for the capacity of the output conveyors. If one of then is capacity negative (i.e. occupied) then it checks for the next one. If both of the conveyors (because for one single color there are only 2 output conveyors) are capacity negative then the dynamic component in the transition point won't move anywhere.
08. The transition point in this [routing_conveyor] is as shown in the following screenshot:



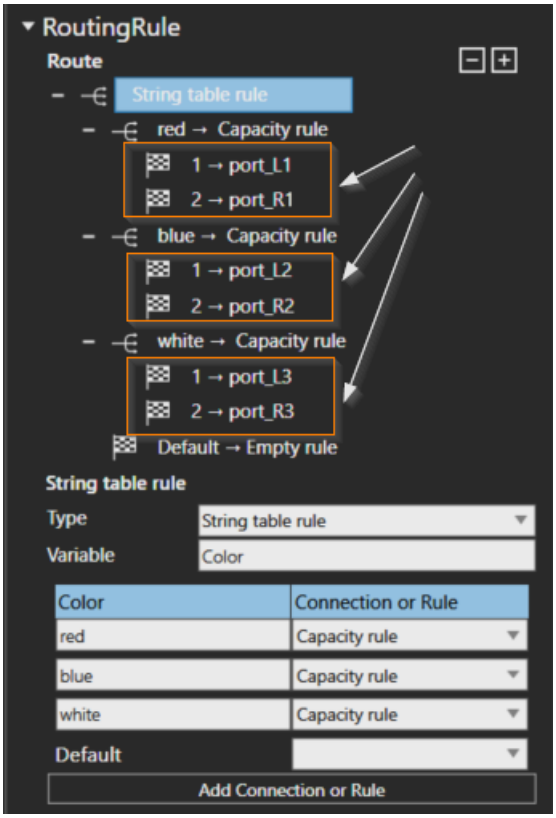


09. The routing rule has been defined as following:-
    - first step - Color Check
    - second step – Capacity Check

Screenshot of the first step [Level One] –



Screenshot of the second step [Level Two] –



10. There are only 2 levels of logic was assigned in this Routing example, but there can be many levels of Rule's that can be assigned for the total routing rule. So the Behaviour of Routing Rule in the software is capable of handling quiet large and complicated routing rules.

11. Note: The way of defining the interfaces for using a conveyor for routing is different from normal interface defining. The designer has to be aware of that. Here is the screenshot showing one interface definition as example: