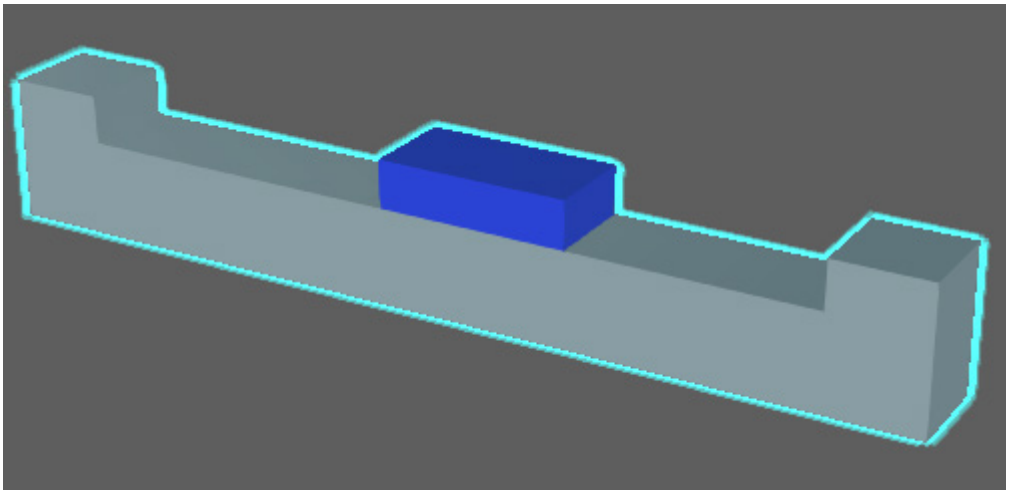
Component Scripting

Visual Components 4.0 | Version: February 28, 2017



A component script allows you to use Python API to create custom behaviors and add logic to a component. For example, you can write a script that manages changes to component properties and what tasks a component performs during a simulation. A component can contain multiple scripts with each script being its own Python Script behavior in the component.

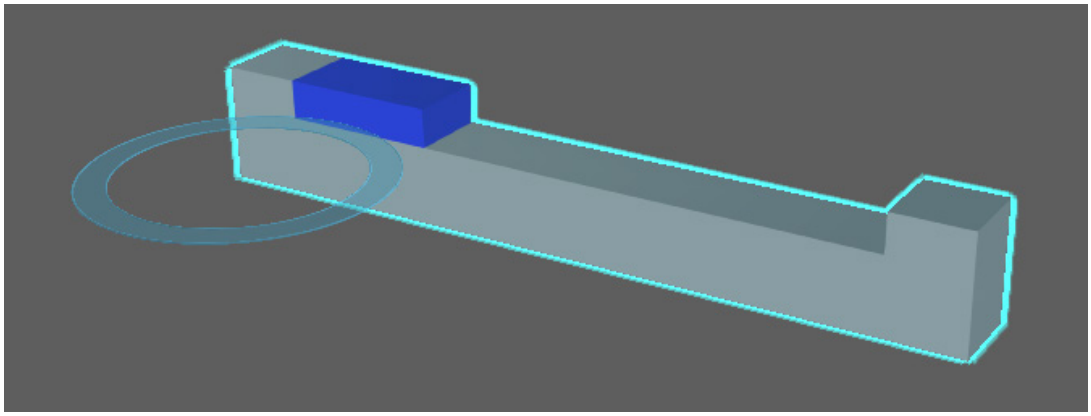
In this tutorial you learn how to create a component script that manages the motions of a servo controller and another script that changes the material of geometry during a simulation. This will involve using methods in the vcScript module to reference the main application and component data.

Support
support@visualcomponents.com

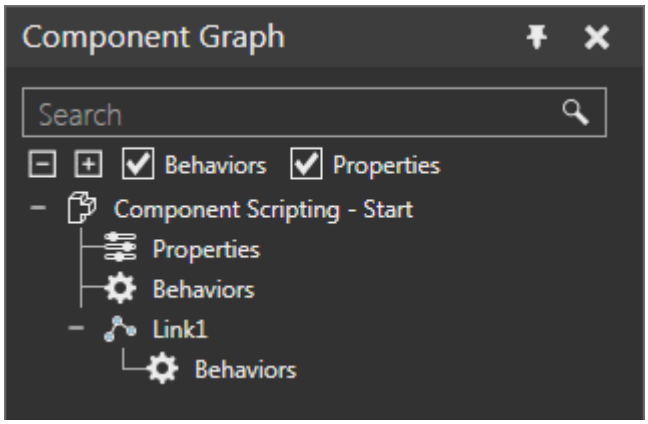
Community
community.visualcomponents.net

Getting Started

1. Open the **ComponentScriptingStart.vcmx** file for this tutorial.



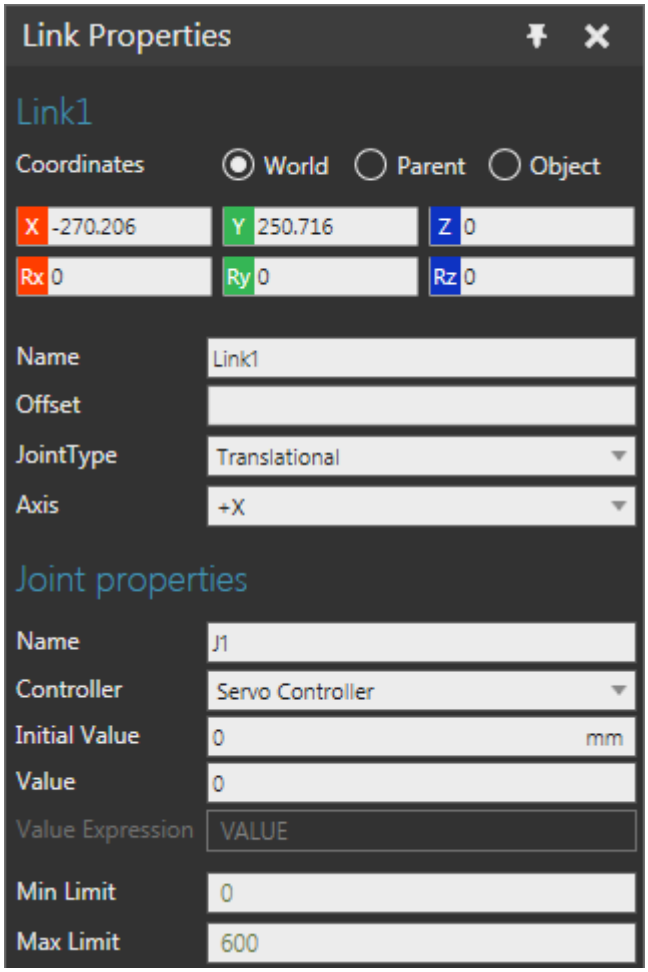
2. Click the **Modeling** tab, and then in the Component Graph panel, select the **Behaviors** and **Properties** check boxes, and then expand the component node tree.



The component has one link containing the geometry of a blue platform. The platform needs to move along the X-axis of its link and be driven by a servo controller. Right now, the component does not have any behaviors and the degree of freedom (DOF) of Link1 is fixed.

3. In the Component Graph panel, select **Link1**, and then in the Link Properties panel, set JointType to **Translational**, and then do all of the following:

   - Set Axis to **+X**.
   - Set Controller to **New Servo Controller**, which will automatically add a new Servo Controller behavior to the root node and assign the joint of Link1 to that controller.
   - Set Min Limit to **0** and Max Limit to **600**.

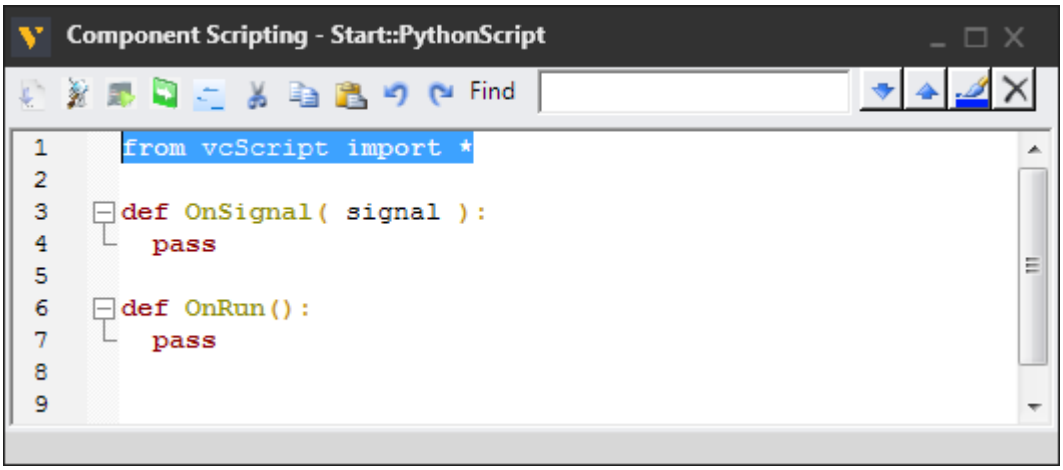


The servo controller needs additional logic to operate during a simulation. For example, the controller needs to know when to move joints, what joints to move, and how far to move them. The logic for the servo controller can be defined in a Python Script behavior.

4. In the Component Graph panel, select the **root node**, and then add a **Python Script** behavior. The script editor will open automatically when you add the behavior.

Basic Script

The initial first line of code for any component script is an import statement that retrieves all methods from the vcScript module.



The **getComponent()** method in vcScript allows you to get an object reference to the component containing the Python Script behavior. This is helpful if you need to get and manipulate other component data, for example behaviors, features and properties.

NOTE! To learn more about the script editor, see "Python Script" in the Behaviors reference guide of your Visual Components 4.0 product Help file.

1. In the script editor, create variables for the component and its servo controller. You can use the component object to call the findBehaviour() method of vcNode to get a behavior by name.

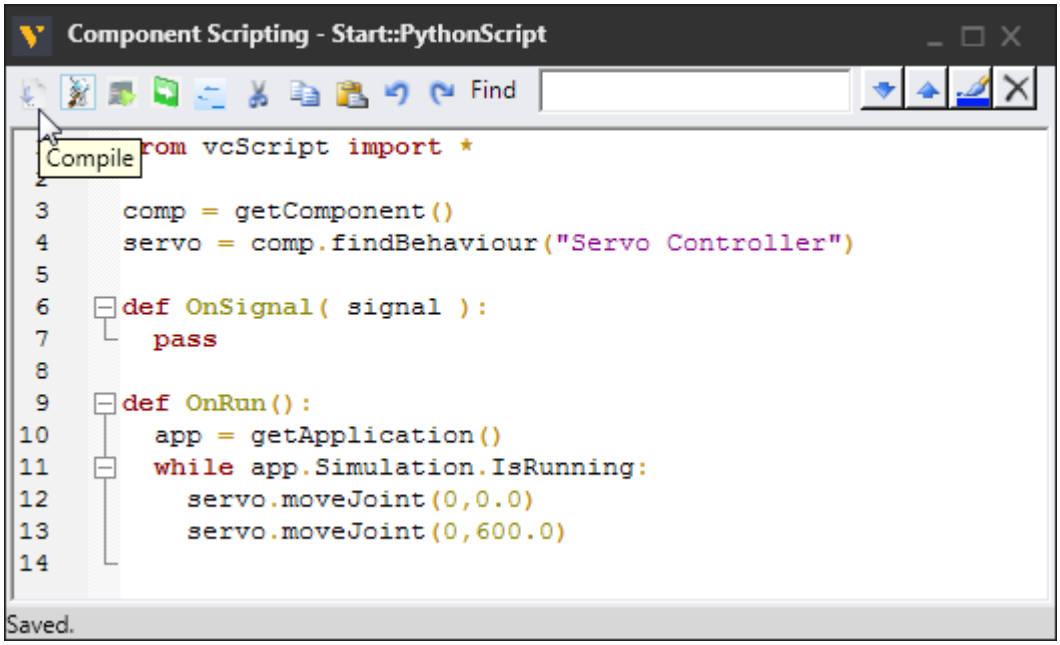
```
from vcScript import *


comp = getComponent()
servo = comp.findBehaviour("Servo Controller")
```

The OnRun event is the main function of a script and is executed during a simulation.

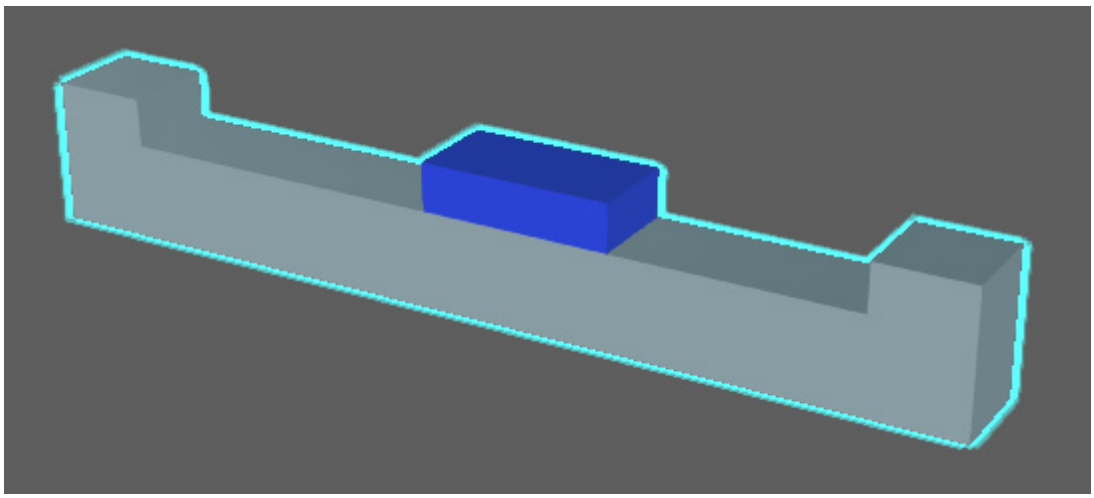
2. In the OnRun event, create a while loop that moves the joint of Link1 from its min to max value and back again. You can use the servo controller object to call the move() or moveJoint() method. To avoid creating an endless loop, you can use the getApplication() method of vcScript to get an object reference for the main application. You could then get a "handle" for the simulation object and make the condition of the while loop to be true as long as the simulation is running.

```
def OnRun():
  app = getApplication()
  while app.Simulation.IsRunning:
    servo.moveJoint(0,0.0)
    servo.moveJoint(0,600.0)
```

3. Compile the code, and then enable **Trace execution**, which will allow you to know which line of code is being executed in the script.



4. Run the simulation, verify the platform moves from one side to the other, and then reset the simulation.



Additional Script

In some cases, you may want to create other scripts in a component that can run independent of one another.

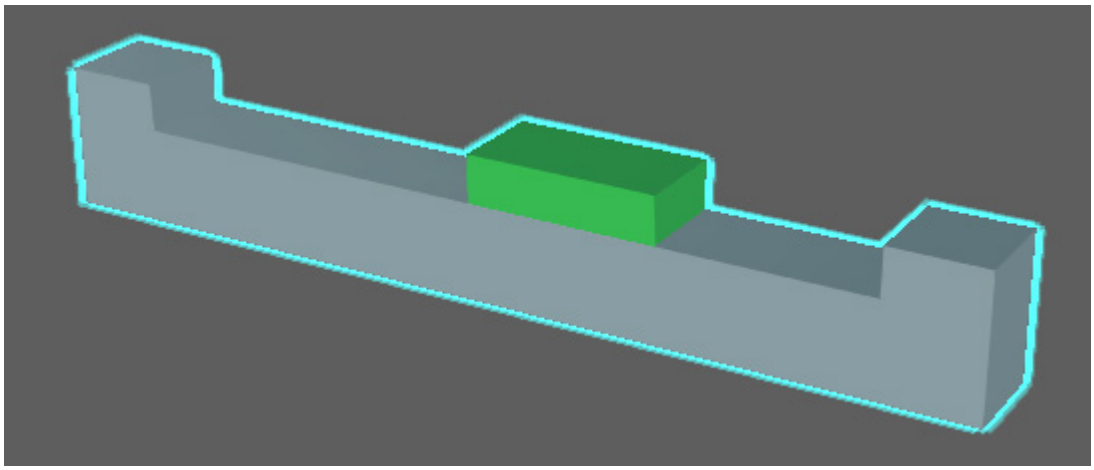
1. Close the script editor, and then add another **Python Script** behavior.

2. In the **PythonScript_2** editor, OnRun event, create a while loop that changes the material of the platform from blue to green. You can use the application object to call the findMaterial() method to get a material by name. The component object can be used to call the findNode() method to get a node by name. In some cases, a feature may already be assigned a material, so you may need to force the geometry of the node to inherit its material. You can use the delay() method of vcScript to delay the execution of a script. This is helpful if you need to toggle something on and off during a simulation.

```
from vcScript import *


def OnSignal( signal ):
  pass


def OnRun():
  app = getApplication()
  blue = app.findMaterial("blue")
  green = app.findMaterial("green")


  comp = getComponent()
  link = comp.findNode("Link1")
  link.MaterialInheritance = VC_MATERIAL_FORCE_INHERIT_NODE


  while app.Simulation.IsRunning:
    delay(1)
    link.NodeMaterial = blue
    delay(1)
    link.NodeMaterial = green
```

3. Compile the code.

4. Run the simulation, verify the platform changes from blue to green, and then reset the simulation.



Review

In this tutorial you learned how to write component scripts that perform tasks during a simulation. You know how to use methods in vcScript to get handles for the main application, component containing the script and its data. You also know how to use the OnRun event to define the main function of a script, which is executed during a simulation.