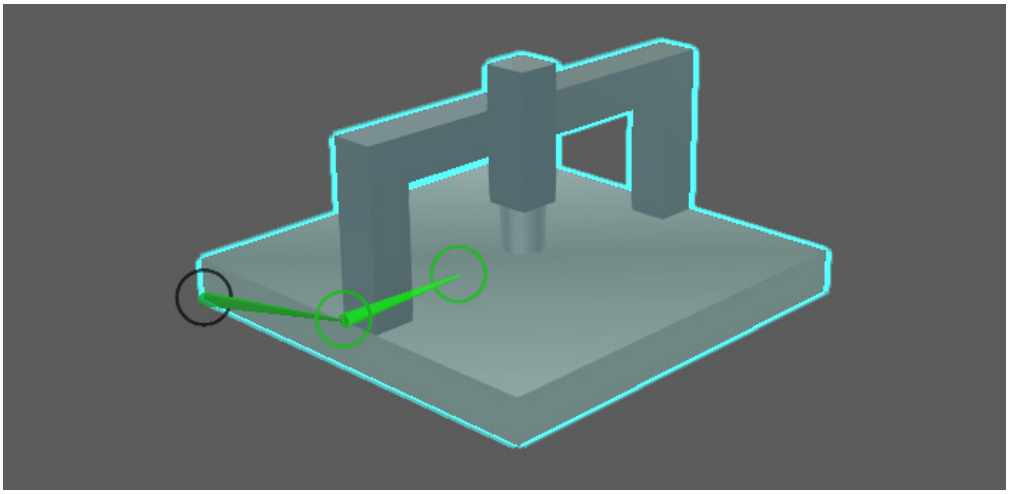
Control Servos

Visual Components 4.0 | Version: March 10, 2017



Python API can be used to control servos during a simulation. For example, you can write a component script that can manage the movements and target values of joints assigned to a servo.

In this tutorial you learn how to:

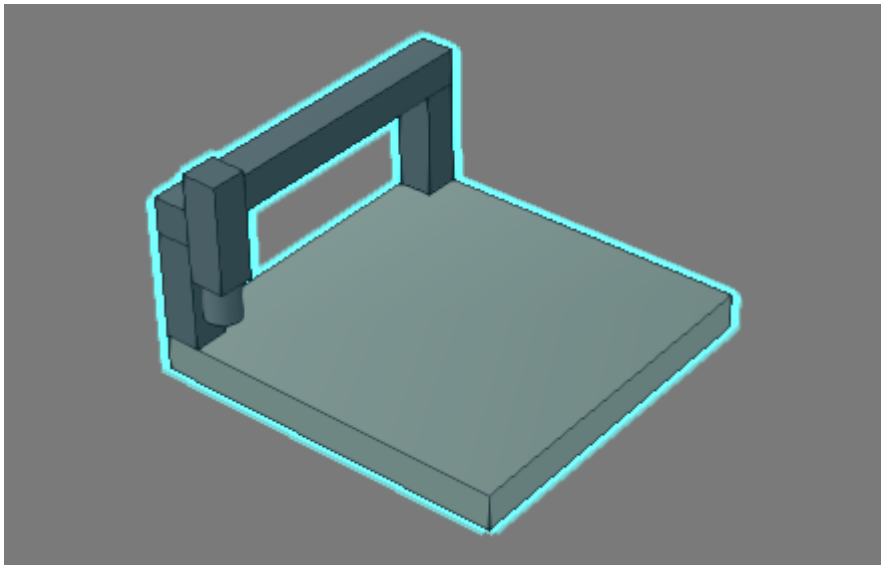
- Get and set joint targets for servo movements.
- Use joints as triggers and conditions for moving other joints.
- Calculate and set cycle times for servo movements.
- Operate a servo in heartbeat mode and use events to modify joint targets.

Support
support@visualcomponents.com

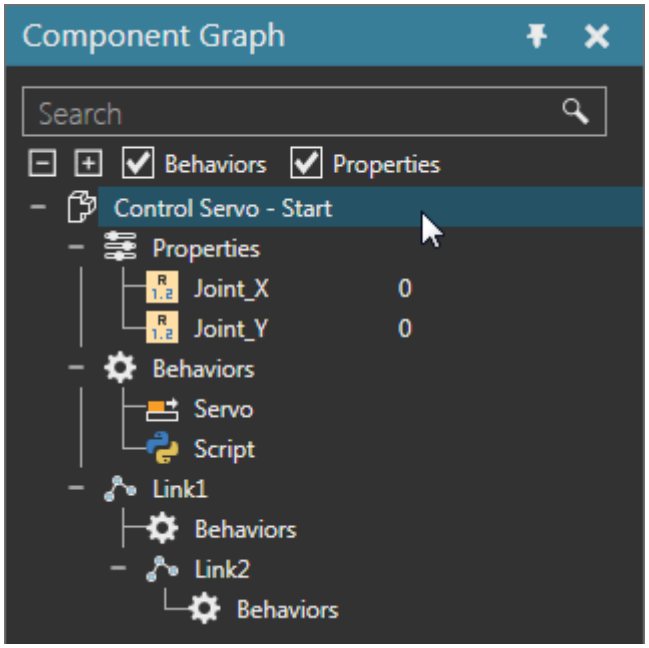
Community
community.visualcomponents.net

Getting Started

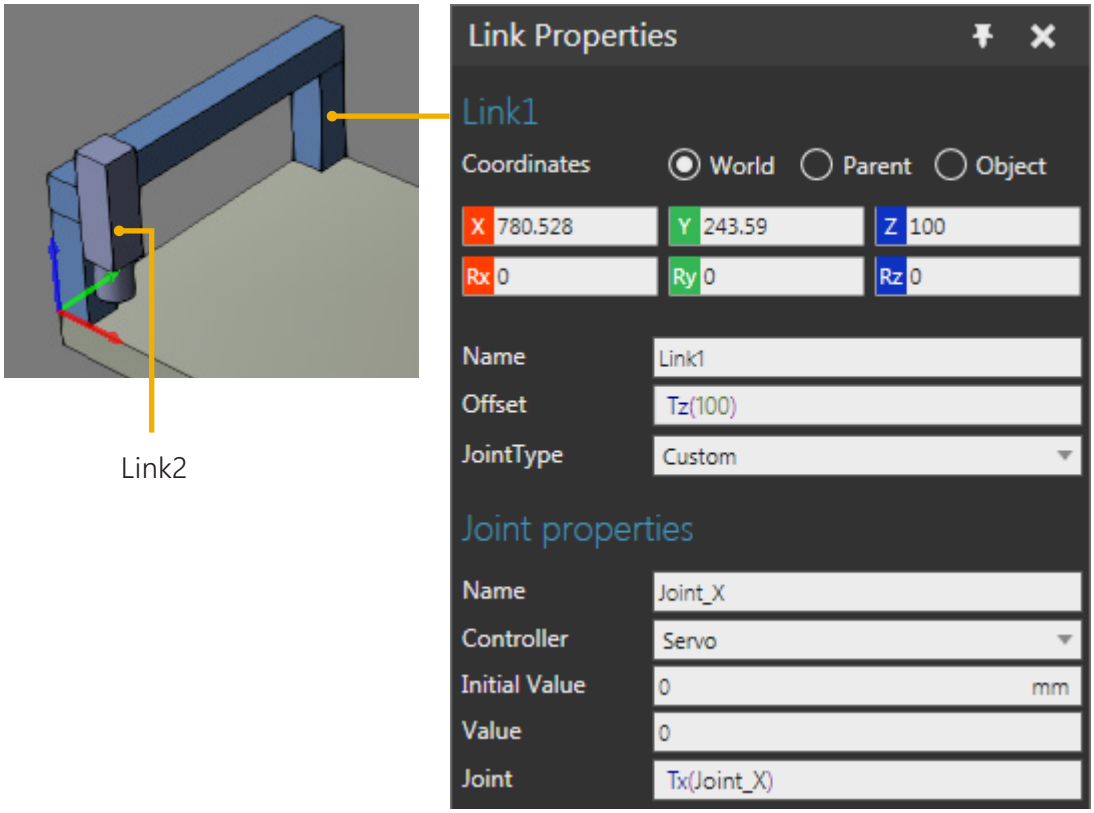
1. Open the **Control Servo - Start.vcmx** file for this tutorial.



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.



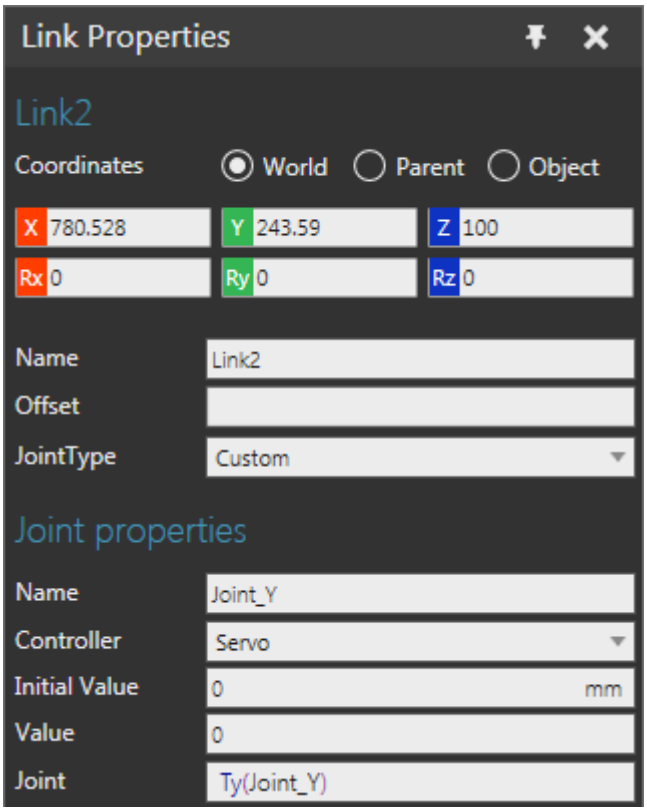
3. Select the **Link1** node, and then access the Link Properties panel.



Link1 has a custom type joint named "Joint_X" which is assigned to a Servo Controller behavior. The degree of freedom (DOF) of Joint_X is a translational movement along its X-axis. When Joint_X is moved it will also move the Link2 node, which is a child node attached to Link1.

4. Select the **Link2** node, and then access the Link Properties panel.

Link2 has a custom type joint named "Joint_Y" which is assigned to a Servo Controller behavior. The DOF of Joint_Y is a translational movement along its Y-axis. Link2 is not offset from its parent node, so the DOF is along the same Y-axis of Link1 coordinate system. When Joint_Y is moved it will only move Link2 and its geometry.



You can use a Python Script behavior to manipulate a servo and its joints during a simulation.

5. In the Component Graph panel, double-click **Script** to access its editor.

6. In the script editor, review lines 1 to 15.

```
from vcScript import *


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


def OnStart():
  for i in servo.Joints:
    print i.Name, i.InitialValue


def OnRun():
  servo.setJointTarget(0,500.0)
  servo.setJointTarget(1,500.0)
  servo.setMotionTime(10.0)
  print servo.calcMotionTime()
  servo.move()
```

In Python API, a Servo Controller behavior is a **vcServoController** object and a joint is a **vcJoint** object. You can create and assign target values for servo joints. That allows you to calculate and manipulate the motion time for reaching those target values. The **move()** method in vcServoController allows you to move joints to target values with optional arguments for passing target values in that call.

7. In the script editor, review lines 17 to 25.

```
def OnRun():
...
  delay(5.0)
  servo.moveJoint(0,250.0)


  trigger = servo.getJointValue(0)
  if trigger <=250.0:
    servo.moveJoint(1,250.0)


  delay(5.0)
  servo.moveImmediate(0.0,0.0)
```

You can also move joints individually by using the **moveJoint()** method or immediately in zero simulation time using the **moveImmediate()** method.

8. In the script editor, review lines 32 to 40.

```
from vcScript import *


comp = getComponent()
servo = comp.findBehaviour("Servo")
...
def myEvent(servo, event):
  global sim
  sim = getSimulation()
  if event == 0:
    print "The servo has started at:", sim.SimTime
  elif event == 2:
    print "The servo has ended at:", sim.SimTime


servo.OnHeartbeat = myEvent
```

A Servo Controller behavior can operate in heartbeat mode, which uses a pulse to drive joints. You can use the **OnHeartbeat** event to monitor the state of the servo and call functions at every heartbeat.

9. Delete **Script**. This will cancel everything in the component script that dealt with the servo, including the removal of event handlers and joint targets.

10. Add a **Python Script** behavior. The script editor will open automatically when you add the behavior.

Joint Targets

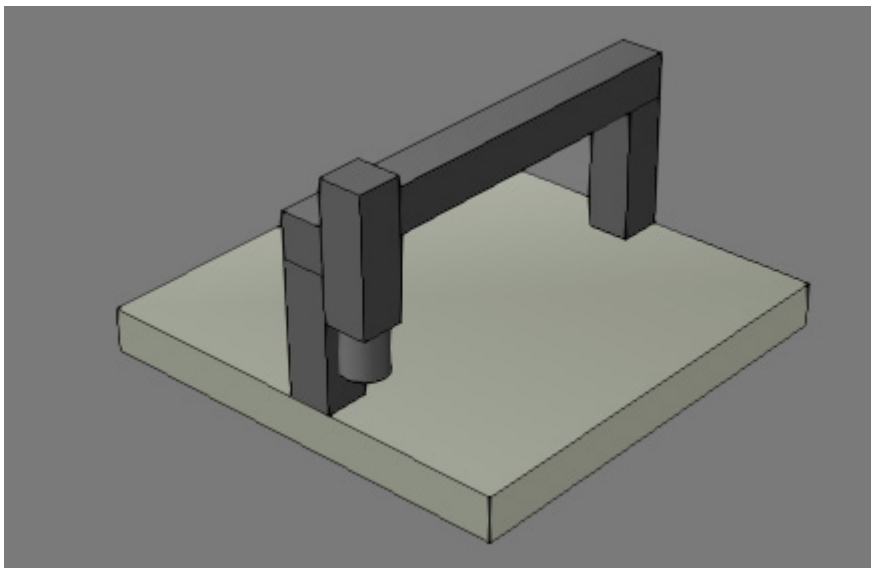
You can instruct a servo to drive a joint to a target value. For example, the **setJointTarget()** method in vcServoController can be used to define a target value for a joint controlled by a servo. You would, however, still need to tell the servo to move the joint.

1. In the script editor, OnRun event, drive Joint_X to a target value of 500.0 and Joint_Y to a target value of 0.0, and then compile the code.

```
from vcScript import *


def OnRun():
  comp = getComponent()
  servo = comp.findBehaviour("Servo")
  servo.setJointTarget(0,500.0)
  servo.setJointTarget(1,0.0)
  servo.move()
```

2. Run the simulation, verify Link1 and its child node Link2 move half way along the length of the component, and then reset the simulation.



Joint Conditions

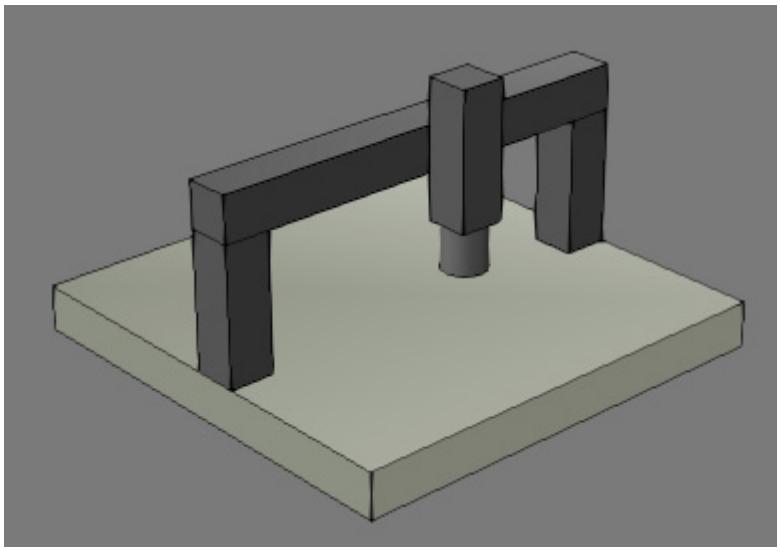
It is possible to use a joint value as a condition for moving another joint. For example, the **getJointValue()** and **getJointTarget()** methods in vcServoController can be used to define a condition for moving connected joints in sequence.

1. In the script editor, OnRun event, create a condition for moving Joint_Y to 500.0 if Joint_X has reached its target value of 500.0, and then compile the code.

```
def OnRun():
  comp = getComponent()
  servo = comp.findBehaviour("Servo")
  servo.setJointTarget(0,500.0)
  servo.setJointTarget(1,0.0)
  servo.move()


  condition(lambda: servo.getJointTarget(0)==servo.getJointValue(0))
  servo.setJointTarget(1,500.0)
  servo.move()
```

2. Run the simulation, verify Link2 moves to the middle of the component after Link1 reaches its target, and the reset the simulation.



Cycle Times

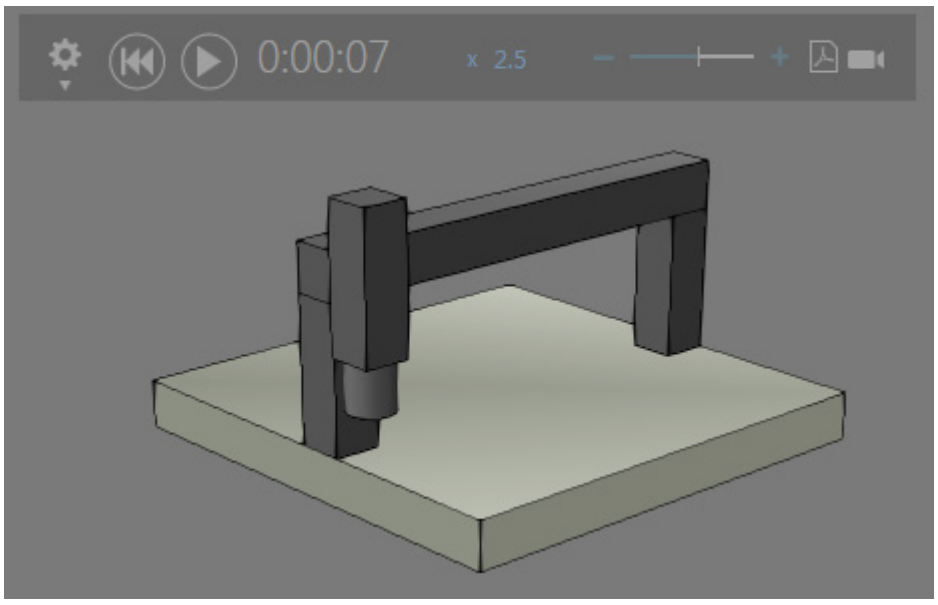
If a servo knows the target values of its joints you can calculate the cycle time of that movement. Another option is to directly set the cylcle time of the servo, thereby scaling joint speeds to complete a movement in that given time. For example, you can use the **calcMotionTime()** and **setMotionTime()** methods in vcServoController to solve for and set motion times of joint movements.

1. In the script editor, OnRun event, calculate the motion time of the first servo movement and add two seconds to it, and then compile the code. It might be helpful to print feedback in the Output panel about the original and new cycle times for that movement.

```
def OnRun():
  comp = getComponent()
  servo = comp.findBehaviour("Servo")
  servo.setJointTarget(0,500.0)
  servo.setJointTarget(1,0.0)

  time = servo.calcMotionTime()
  print "Original cycle time", time
  time += 2
  servo.setMotionTime(time)
  print "New cycle time: ", servo.calcMotionTime()
  servo.move()
...
```

2. Run the simulation, verify the first movement of the servo takes about 7.2 seconds, and then reset the simulation.



Joint Movements

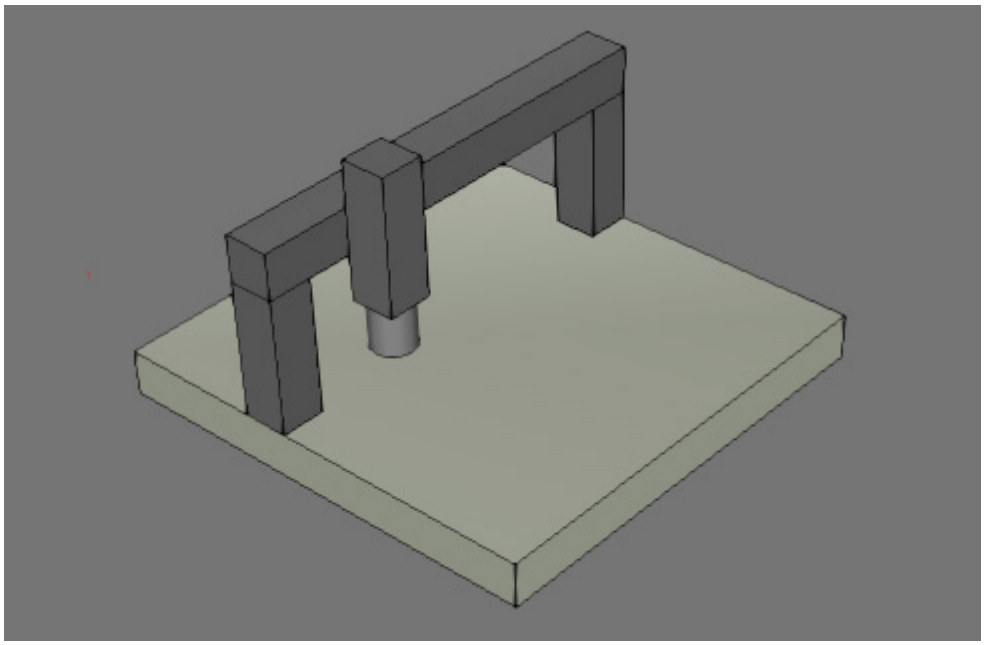
It is possible for a servo to move joints without having to first set target values. For example, you can use the **moveImmediate()** method in vcServoController to move joints to target values in zero simulation time. Another option is to use the **moveJoint()** method to drive a specific joint. Finally, you can pass target values for joints as optional arguments in the **move()** method, but the values would need to be given in the order that joints are listed in the servo.

1. In the script editor, modify the OnRun event to move all servo joints to 0.0 immediately, the first joint to 800.0 and then the second joint to 500.0, and then all joints to 0.0 in that order, and then compile the code.

```
from vcScript import *


def OnRun():
  comp = getComponent()
  servo = comp.findBehaviour("Servo")
  #four movements
  servo.moveImmediate(0,0)
  servo.moveJoint(0,800.0)
  servo.moveJoint(1,500.0)
  servo.move(0,0)
```

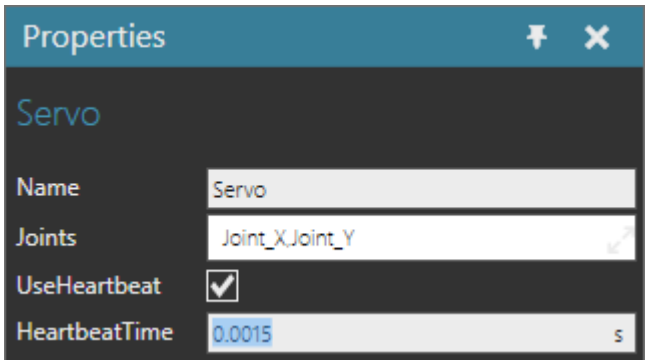
2. Run the simulation, verify the servo executes the movements correctly and in the right order, and then reset the simulation. Notice in the last movement that both joints move at the same time because the servo is driving both joints to zero value.



Heartbeat Mode

A servo controller can operate in heartbeat mode, thereby allowing you to handle servo events and simulate pulse-driven movements.

1. In the Component Graph panel, select **Servo**, and then in the Properties panel, select the **UseHeartbeat** check box and set HeartbeatTime to **0.0015** or 1.5ms.



2. In the script editor, create an event handler for the OnHeartbeat event that sets the next joint targets of the servo. In this case, the event handler is used to resume the OnRun event and a global targets variable is used to store target values.

```
from vcScript import *


targets = None
def addNewTarget(servo,event_type):
  global targets
  if event_type == VC_CONTROLLER_END:
    if servo.getJointTarget(0) == 0 and servo.getJointTarget(1) == 0:
      targets = (500,0)
    elif servo.getJointTarget(0) == 500 and servo.getJointTarget(1) == 0:
      if len(targets) < 3:
        targets = (500,500)
      else:
        targets = (0,0)
    elif servo.getJointTarget(0) == 500 and servo.getJointTarget(1) == 500:
      targets = (500,0,-1)
    for i in range(servo.JointCount):
      servo.setJointTarget(i,targets[i])
      resumeRun()


comp = getComponent()
servo = comp.findBehaviour("Servo")
servo.OnHeartbeat = addNewTarget
```

3. Modify the OnRun event to use a while loop for driving servo joints, and then compile the code. In this case, joint targets are assigned at the beginning of the OnRun event in order to trigger the VC_CONTROLLER_END event. That is, the first movement of the servo is a warm up and all subsequent movements are event-driven.

```
def OnRun():
  for i in range(servo.JointCount):
    servo.setJointTarget(i,0)
  while True:
    servo.move()
    suspendRun()
```

4. Run the simulation, verify the servo executes a loop of movements that are event-driven, and then reset the simulation.

Review

In this tutorial you learned how to control a servo and its joints. You know how to set joint target values, initiate movements, and calculate and set cycle times for those movements. You also know how to use joints to trigger the movements of other joints. Finally, you know how to use events to set joint targets and control servo movements in a process.