**Support for 64-bit Linux Mint 18**

**Why Transitions?**

Transitions is an object-based, versatile state machine library in Python. We chose to use this tool alongside STK because STK allows for Python integration and customization of the COLLADA files within the scenario.

**Installing Transitions**

1. Install Python 3.
   1. If you do not have it, get it here: <https://www.python.org/downloads/>
2. Open a Windows command prompt
3. Enter this into the command prompt: pip install transitions
   1. Machine path variables are different; depending on yours, you may need to enter one of these alternate commands:

py pip install transitions

OR

py -m pip install transitions

1. Once you get the message that Transitions was successfully installed, you can start using it to make state machines or edit existing ones

**Importing Transitions**

1. Open a python file.
2. At the top, have: from transitions import Machine

**Initializing a State Machine**

1. Each state machine is an object, so it must be defined as a class
   1. In the below example, this is “class solar\_panelNode(object):”
2. In the class, have a list of strings that serve as states that the component can be in
   1. In the example, this is the “states” list containing: on, off, engage, collecting light, not collecting light
3. Set up a constructor for the object. This is automatically called whenever you create an object
   1. In the example, this is the “def \_\_init\_\_(self, name)” function
4. Set any variables using “self.X” notation
   1. self.name and self.efficiency are examples of this
5. Initialize the state machine by “self.machine = Machine(model=self, states=”STATELIST”, initial=”STATE”) where “states” is equal to the list you initialized in step 2 and initial is a string in that list
   1. In the below example, “states” = the states list and the initial state = ‘off’
6. Add different transitions to the machine by using “self.machine.add\_transition(trigger=”FUNCTIONNAME”, source=’FIRST STATE’, dest=’NEXT STATE’) where trigger can be equal to any string or existing function.

Example state machine:

class solar\_panelNode(object):

states = ['on', 'off', 'engage', 'collecting light', 'not collecting light']

def \_\_init\_\_(self, name):

#Set node name and efficiency from global variable

self.name = name

self.efficiency = EFFICIENCY

self.machine = Machine(model=self, states=solar\_panelNode.states, initial='off')

self.machine.add\_transition(trigger='turnOn', source='off', dest='on')

self.machine.add\_transition(trigger='inSun', source='on', dest='collecting light')

self.machine.add\_transition(trigger='inSun', source='not collecting light', dest='collecting light')

self.machine.add\_transition(trigger='inEclipse', source='on', dest='not collecting light')

self.machine.add\_transition(trigger='inEclipse', source='collecting light', dest='not collecting light')

**Next thing**

1. A