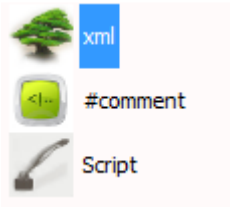
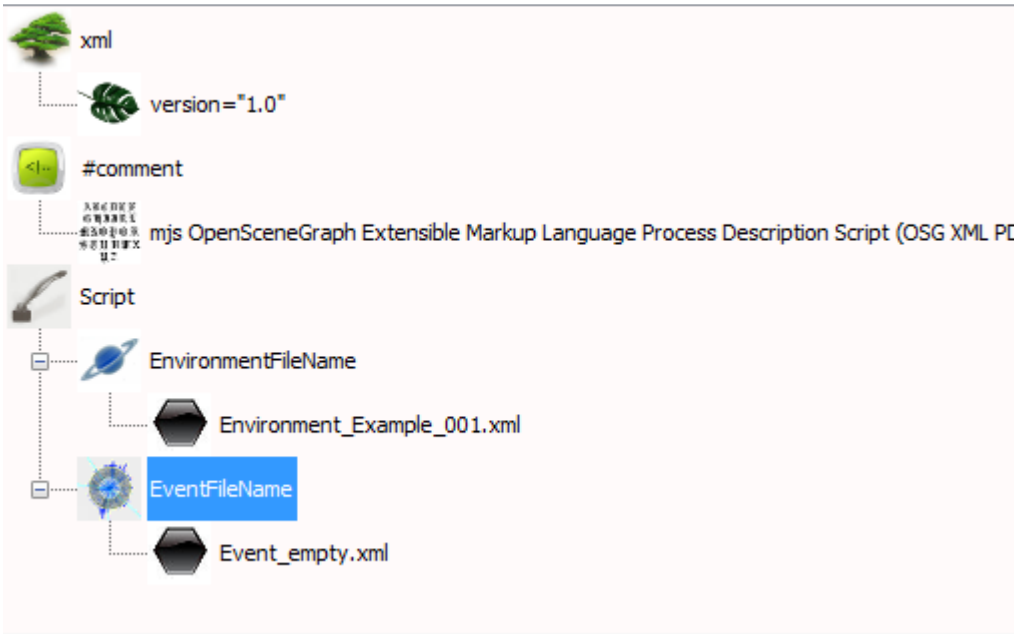
Script_Example_001.xml



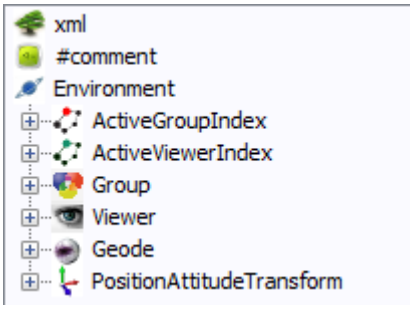
Click on the xml and comment icons to open up child items in tree control

Note: Environment and Event file names

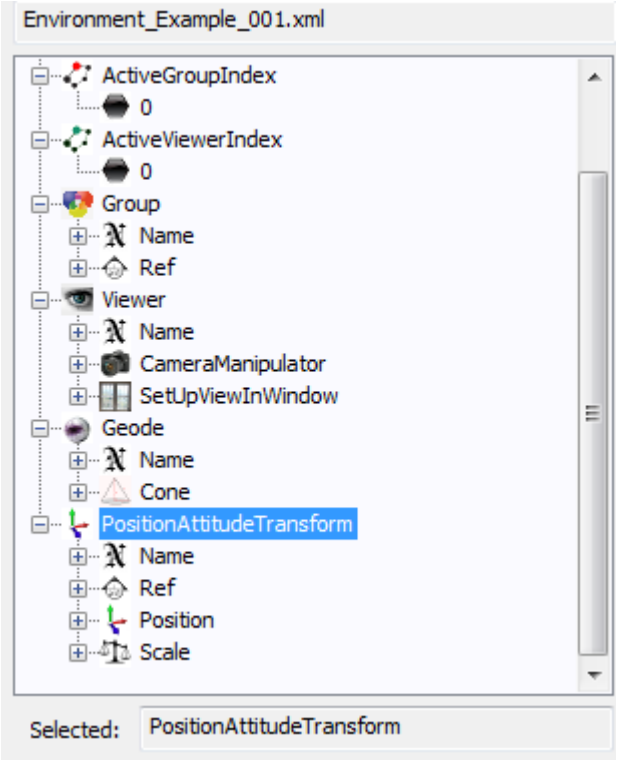


Environment_Example_001.xml

Click on the xml and comment icons again to minimise the branch in tree control. Click on Environment to see the entities contained in the environment.

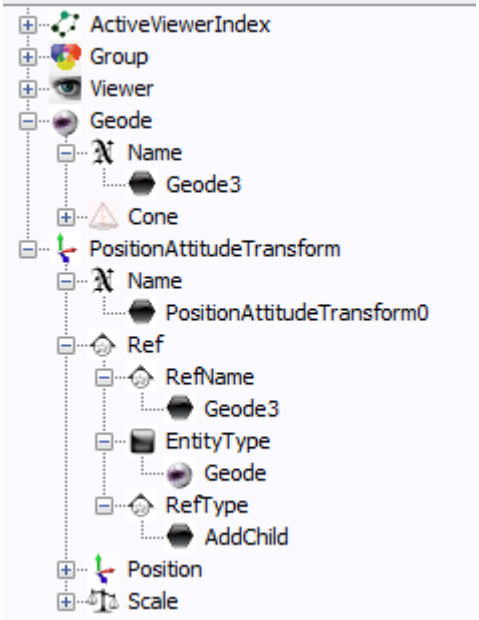


Note that there is one Viewer, one Group and that the ActiveGroupIndex and ActiveViewerIndex are set to reference these.



To render a cone we can create a Geode and a PositionAttitudeTransform (PAT)

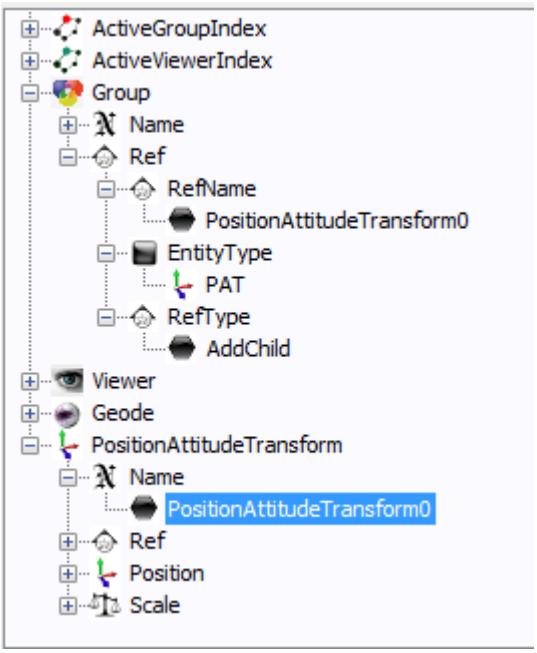
See that the PositionAttitudeTransform (PAT) entity contains a reference to the Geode. Specifically the Ref object has a Geode *EntityType* and *RefName* correlates to the Name of the Geode.



After the engine parses the script it links entities together as described by references

```
PAT_iter->mp_PAT->addChild(mv_geode[ref_iter->m_index].mp_geode);
```

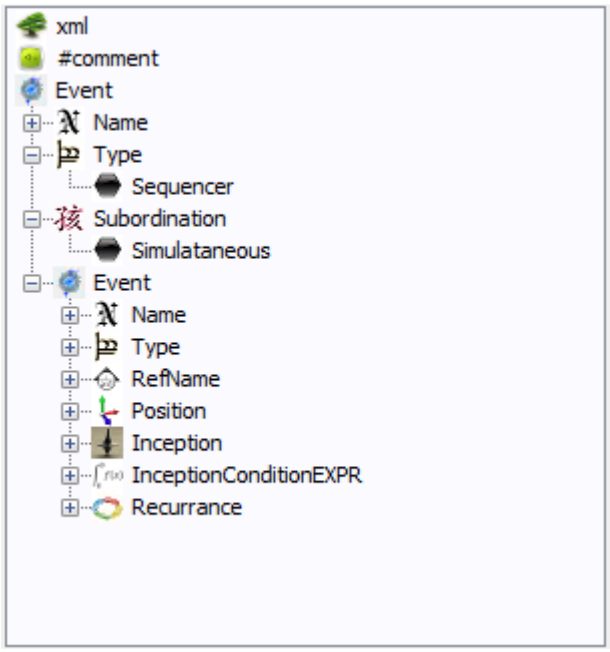
The Viewer entity can render a Group. The Geode or the PositionAttitudeTransform (that references the Geode) needs to be reference by a Group that can be rendered. In this case we add a Ref to a Group object. See that the reference *EntityType* is PositionAttitudeTransform and the *RefName* correlates to the Name of the PositionAttitudeTransform.



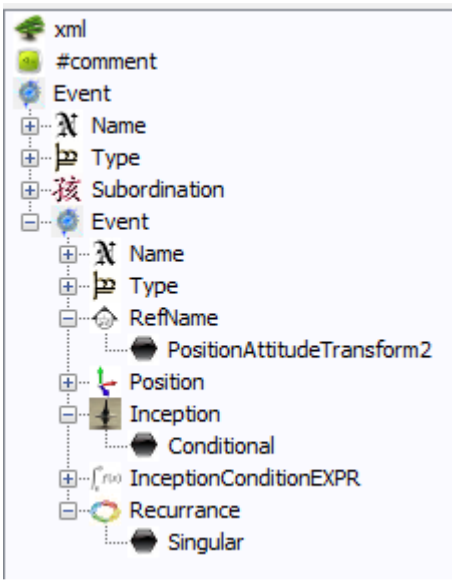

```
group_iter->mp_group->addChild(mv_PAT[ref_iter->m_index].mp_PAT);
```

```
mv_viewer[m_active_viewer_index].mp_viewer->setSceneData(mv_group[m_active_group_index].mp_group);
```

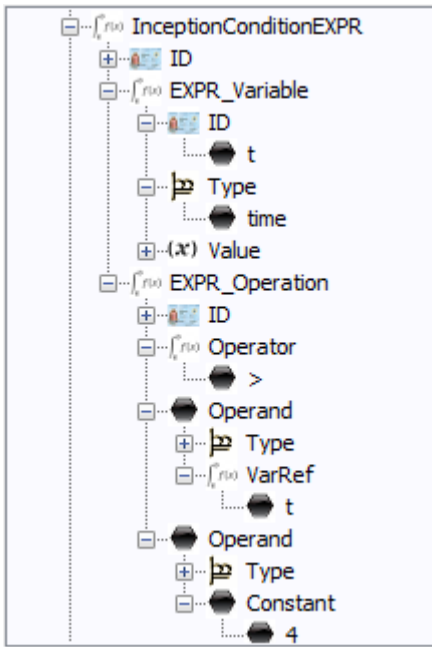
In this Event script the parent or root event *Type* is *Sequencer* which is an event which manages the update of one or many subordinate events. In this case the *Subordination* is *Simultaneous* which means that all of the child events contained in this object are updated when the parent is updated. Subordinate events may also be updated sequentially or randomly.



By default the event inception is spontaneous so it is not necessary to add an Inception element to the script. Likewise the event recurrence is singular. In this case we want an event that updates when a condition is met and we want it to update just once. The child event an *UpdatePosition* Type event so we need to include a *RefName* element that contains the name of the PAT to be updated.



The Inception condition is defined with an expression or EXPR element. Within an *Event* object the specific script element to use is *InceptionConditionEXPR*



Here we can see that the expression consists of one variable and one operation. The variable is defined as being of Type time. By defining the Type as time and the ID as t we are creating a variable t that is the elapsed time (in seconds) since the engine process begun. The operation *Operator* is a greater than symbol ">". The > Operator is a binary operation and it requires two *Operands* (as opposed to a unary or ternary operations which require 1 and 3 Operands). The first Operand (Op1) is considered to be on the left side of the Operator and the second Operand (Op2) is on the right. So we have the Operation Op2 > Op1 and thus the expression is t > 4. When the expression is true or equal to 1 our inception condition is met. As the recurrence is singular the event updates just the once and sets the position of the PAT before it is destroyed.

```
<?xml version="1.0"?>
<!--mjs-->
<Event>
  <Name>$Sequencer</Name>
  <Type>Sequencer</Type>
  <Subordination>Sequential</Subordination>
  <Event>
    <Name>$Name</Name>
    <Type>AlterPosition</Type>
    <RefName>PositionAttitudeTransform0</RefName>
    <Inception>Conditional</Inception>
    <InceptionConditionEXPR>
      <ID>Inception</ID>
      <EXPR_Variable>
        <ID>x</ID>
        <Type>time</Type>
      </EXPR_Variable>
      <EXPR_Operation>
        <ID>F</ID>
        <Operator>></Operator>
        <Operand>
          <Type>Variable</Type>
          <VarRef>x</VarRef>
        </Operand>
        <Operand>
          <Type>Constant</Type>
          <Constant>8</Constant>
        </Operand>
      </EXPR_Operation>
    </InceptionConditionEXPR>
    <Recurrance>Singular</Recurrance>
    <Position>
      <x>0</x>
      <y>0</y>
      <z>3</z>
    </Position>
  </Event>
</Event>
```