This is a test of x3d.py (by Masaki Aono , Augst 26th, 2019)

In [1]:

```
import x3d
```

```
x3d package loaded, have fun with X3D Graphics!
```

Hello World example

In [2]:

```
###############################################
#
# Now available: deployment of an alpha 'x3d' python package on PyPi for import.
#   This approach greatly simplifies deployment and use, avoiding extra setup.
#   https://pypi.org/project/x3d
#   https://twitter.com/Web3DConsortium/status/1154449868846297088
#
# Installation:
#       pip install x3d
# or
#       python -m pip install x3d
#
# TODO Add documentation and stylesheet parameters for enabling/disabling these options.
#
# Developer options for loading x3d package:
#    import x3d        # pythonic approach to subclass x3d package, elements require x3d.* prefi
x
# or
#    from x3d import * # "polluting" version of import that avoids x3d.* prefixes

import x3d

###############################################

newModel=x3d.X3D(profile='Immersive',version='3.3',
  head=x3d.head(
    children=[
    x3d.meta(content='HelloWorld.x3d',name='title'),
    x3d.meta(content='Simple X3D scene example: Hello World!',name='description'),
    x3d.meta(content='30 October 2000',name='created'),
    x3d.meta(content='23 September 2017',name='modified'),
    x3d.meta(content='Don Brutzman',name='creator'),
    x3d.meta(content='HelloWorld.tall.png',name='Image'),
    x3d.meta(content='http://en.wikipedia.org/wiki/Hello_world',name='reference'),
    x3d.meta(content='https://en.wikipedia.org/wiki/Hello#.22Hello.2C_World.22_computer_program'
,name='reference'),
    x3d.meta(content='https://en.wikipedia.org/wiki/"Hello,_World!"_program',name='reference'),
    x3d.meta(content='http://en.wikibooks.org/w/index.php?title=Computer_Programming/Hello_worl
d',name='reference'),
    x3d.meta(content='http://www.HelloWorldExample.net',name='reference'),
    x3d.meta(content='http://www.web3D.org',name='reference'),
    x3d.meta(content='http://www.web3d.org/realtime-3d/news/internationalization-x3d',name='refe
rence'),
    x3d.meta(content='http://www.web3d.org/x3d/content/examples/HelloWorld.x3d',name='reference'
),
    x3d.meta(content='http://X3dGraphics.com/examples/X3dForAdvancedModeling/HelloWorldScenes',n
ame='reference'),
    x3d.meta(content='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter01TechnicalOvervie
w/HelloWorld.x3d',name='identifier'),
    x3d.meta(content='http://www.web3d.org/x3d/content/examples/license.html',name='license'),
    x3d.meta(content='X3D-Edit 3.3, https://savage.nps.edu/X3D-Edit',name='generator'),
    #  Alternate encodings: VRML97, X3D ClassicVRML Encoding, X3D Compressed Binary Encoding (CB
E), X3DOM, JSON
    x3d.meta(content='HelloWorld.wrl',name='reference'),
    x3d.meta(content='HelloWorld.x3dv',name='reference'),
    x3d.meta(content='HelloWorld.x3db',name='reference'),
    x3d.meta(content='HelloWorld.xhtml',name='reference'),
    x3d.meta(content='HelloWorld.json',name='reference')]),
  Scene=x3d.Scene(
    #  Example scene to illustrate X3D nodes and fields (XML elements and attributes)
    children=[
    x3d.WorldInfo(title='Hello World!'),
    x3d.Group(
      children=[
      x3d.Viewpoint(DEF='ViewUpClose',centerOfRotation=(0,-1,0),description='Hello world!',posit
ion=(0,-1,7)),
      x3d.Transform(rotation=(0,1,0,3),
        children=[
        x3d.Shape(
          geometry=x3d.Sphere(),
          appearance=x3d.Appearance(
            material=x3d.Material(DEF='MaterialLightBlue',diffuseColor=(0.1,0.5,1)),
            texture=x3d.ImageTexture(DEF='ImageCloudlessEarth',url=["earth-topo.png","earth-top
o.jpg","earth-topo-small.gif","http://www.web3d.org/x3d/content/examples/Basic/earth-topo.png",
"http://www.web3d.org/x3d/content/examples/Basic/earth-topo.jpg","http://www.web3d.org/x3d/conte
nt/examples/Basic/earth-topo-small.gif"])))]),
      x3d.Transform(translation=(0,-2,0),
        children=[
        x3d.Shape(
          geometry=x3d.Text(DEF='TextMessage',string=["Hello","world!"],
            fontStyle=x3d.FontStyle(justify=["MIDDLE","MIDDLE"])),
          appearance=x3d.Appearance(
            material=x3d.Material(USE='MaterialLightBlue')))])])])
) # X3D model complete

###############################################
# Self-test diagnostics
###############################################

if         x3d.metaDiagnostics(newModel):
    print (x3d.metaDiagnostics(newModel))

print ("python load successful HelloWorld.py")
```

```
meta
python load successful HelloWorld.py
```

In [3]:

```
print(newModel)
```

```
X3D(head=head(children=[meta(content='HelloWorld.x3d',name='title'), meta(content
='Simple X3D scene example: Hello World!',name='description'), meta(content='30 Oc
tober 2000',name='created'), meta(content='23 September 2017',name='modified'), me
ta(content='Don Brutzman',name='creator'), meta(content='HelloWorld.tall.png',name
='Image'), meta(content='http://en.wikipedia.org/wiki/Hello_world',name='referenc
e'), meta(content='https://en.wikipedia.org/wiki/Hello#.22Hello.2C_World.22_comput
er_program',name='reference'), meta(content='https://en.wikipedia.org/wiki/"Hello,
_World!"_program',name='reference'), meta(content='http://en.wikibooks.org/w/inde
x.php?title=Computer_Programming/Hello_world',name='reference'), meta(content='htt
p://www.HelloWorldExample.net',name='reference'), meta(content='http://www.web3D.o
rg',name='reference'), meta(content='http://www.web3d.org/realtime-3d/news/interna
tionalization-x3d',name='reference'), meta(content='http://www.web3d.org/x3d/conte
nt/examples/HelloWorld.x3d',name='reference'), meta(content='http://X3dGraphics.co
m/examples/X3dForAdvancedModeling/HelloWorldScenes',name='reference'), meta(conten
t='http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter01TechnicalOverview/Hel
loWorld.x3d',name='identifier'), meta(content='http://www.web3d.org/x3d/content/ex
amples/license.html',name='license'), meta(content='X3D-Edit 3.3, https://savage.n
ps.edu/X3D-Edit',name='generator'), meta(content='HelloWorld.wrl',name='referenc
e'), meta(content='HelloWorld.x3dv',name='reference'), meta(content='HelloWorld.x3
db',name='reference'), meta(content='HelloWorld.xhtml',name='reference'), meta(con
tent='HelloWorld.json',name='reference')]),Scene=Scene(children=[WorldInfo(title
='Hello World!'), Group(children=[Viewpoint(DEF='ViewUpClose',centerOfRotation=(0,
-1, 0),description='Hello world!',position=(0, -1, 7)), Transform(rotation=(0, 1,
0, 3),children=[Shape(appearance=Appearance(material=Material(DEF='MaterialLightBl
ue',diffuseColor=(0.1, 0.5, 1)),texture=ImageTexture(DEF='ImageCloudlessEarth',url
=[earth-topo.png, earth-topo.jpg, earth-topo-small.gif, http://www.web3d.org/x3d/c
ontent/examples/Basic/earth-topo.png, http://www.web3d.org/x3d/content/examples/Ba
sic/earth-topo.jpg, http://www.web3d.org/x3d/content/examples/Basic/earth-topo-sma
ll.gif])),geometry=Sphere())]), Transform(translation=(0, -2, 0),children=[Shape(a
ppearance=Appearance(material=Material(USE='MaterialLightBlue')),geometry=Text(DEF
='TextMessage',string=[Hello, world!],fontStyle=FontStyle(justify=[MIDDLE, MIDDL
E]))])])]))
```

In [4]:

```
Scene25 = x3d.Scene()
```

In [5]:

```
WorldInfo26 = x3d.WorldInfo()
```

In [6]:

```
Viewpoint28 = x3d.Viewpoint()
```

In [7]:

```
dir(Viewpoint28)
```

Out[7]:

```
['DEF',
 'FIELD_DECLARATIONS',
 'IS',
 'USE',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__name__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repl__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'centerOfRotation',
 'class_',
 'description',
 'fieldOfView',
 'jump',
 'metadata',
 'orientation',
 'position',
 'retainUserOffsets',
 'specificationUrl',
 'tooltip']
```

Smoke Test

In [8]:

```
##################################################
#
# Now available: deployment of an alpha 'x3d' python package on PyPi for import.
#   This approach greatly simplifies deployment and use, avoiding extra setup.
#   https://pypi.org/project/x3d
#   https://twitter.com/Web3DConsortium/status/1154449868846297088
#
# TODO Add documentation and stylesheet parameters for enabling/disabling these options.
#
# Developer options for loading x3d package:
#    import x3d       # pythonic approach to subclass x3d package, elements require x3d.* prefi
x
# or
#    from x3d import * # "polluting" version of import that avoids need to prepend x3d.* prefixe
s

print ("===================")
print ("Importing local development copy of X3D package:")
print ("  from x3d import *")

# import x3d

from x3d import * # "polluting" version of import that avoids need to prepend "x3d." prefix
print ("===================")

##################################################

print ("PythonX3dSmokeTests:")
test = SFBool()
test = SFBool(True)
test = SFBool(False)
test = SFBool(value=True)
test = SFBool(value=False)
test = SFBool('True')
test = SFBool('False')
test = SFBool('true')
test = SFBool('false')
#test.value = 'invalid'
test.value = True
print ("SFBool test       =", test)
print ("SFBool test.value =", test.value)

test = MFBool()
test = MFBool([True,False,'True','False','true','false']) # ,None is not valid
#test.value = 'invalid'
#test.value = [False,True,]
# test = MFBool( True,False ) # TODO
print ("MFBool test       =", test)
print ("MFBool test.value =", test.value)

test = SFInt32()
test = SFInt32(-1)
#test.value = 'invalid'
test.value = 1
print ("SFInt32 test       =", test)
print ("SFInt32 test.value =", test.value)

test = MFInt32()
test = MFInt32([0,1,2])
```

```
# test = MFInt32( 1,2,3,4 ) # TODO
#test.value = 'invalid'
test.value = [0,1,2,3,4,5]
print ("MFInt32 test       =", test)
print ("MFInt32 test.value =", test.value)

test = SFFloat()
test = SFFloat(-1)
test.value = 1
print ("SFFloat test       =", test)
print ("SFFloat test.value =", test.value)

test = MFFloat()
test = MFFloat([0,1,2])
# test = MFFloat( 1,2,3,4 ) # TODO
print ("MFFloat test       =", test)
print ("MFFloat test.value =", test.value)

test = SFDouble()
test = SFDouble(-1)
test.value = 1
print ("SFDouble test       =", test)
print ("SFDouble test.value =", test.value)

test = MFDouble()
test = MFDouble([0,1,2])
# test = MFDouble( 1,2,3,4 ) # TODO
print ("MFDouble test       =", test)
print ("MFDouble test.value =", test.value)

test = SFString()
test = SFString("test constructor")
test.value = 'test setter'
print ("SFString test       =", test)
print ("SFString test.value =", test.value)

test = MFString()
# test = MFString( "hello", "test" ) # TODO
test = MFString(["test", "constructor"]) # comma necessary or python catenates strings
test.value = ['test', 'setter']
print ("MFString test       =", test)
print ("MFString test.value =", test.value)
# - - - - - - -
test = SFVec2f()
test = SFVec2f((-1, -2)) # commas required
test.value = (1, 2)      # commas required
print ("SFVec2f test       =", test)
print ("SFVec2f test.value =", test.value)

test = MFVec2f()
test = MFVec2f([(-1,-2),(-3,-4)]) # commas required
test.value =   [(0,1),(2,3)]  # commas required
# test = MFVec2f( 1,2,3,4 ) # TODO
print ("MFVec2f test       =", test)
print ("MFVec2f test.value =", test.value)

test = SFVec2d()
test = SFVec2d((-1, -2)) # commas required
test.value = (1, 2)      # commas required
print ("SFVec2d test       =", test)
print ("SFVec2d test.value =", test.value)
```

```
test = MFVec2d()
test = MFVec2d([(-1,-2),(-3,-4)]) # commas required
test.value =   [(0,1),(2,3)]  # commas required
# test = MFVec2d( 1,2,3,4 ) # TODO
print ("MFVec2d test       =", test)
print ("MFVec2d test.value =", test.value)
# - - - - - - -
test = SFVec3f()
test = SFVec3f((-1, -2, -3)) # commas required
test.value = (1, 2, 3)       # commas required
print ("SFVec3f test       =", test)
print ("SFVec3f test.value =", test.value)

test = MFVec3f()
test = MFVec3f([(-1,-2,-3),(-4,-5,-6)]) # commas required
test.value =   [(0,1,2),(3,4,5)]  # commas required
# test = MFVec3f( 1,2,3,4,5,6 ) # TODO
print ("MFVec3f test       =", test)
print ("MFVec3f test.value =", test.value)
# - - - - - - -
test = SFVec3d()
test = SFVec3d((-1, -2, -3)) # commas required
test.value = (1, 2, 3)       # commas required
print ("SFVec3d test       =", test)
print ("SFVec3d test.value =", test.value)

test = MFVec3d()
test = MFVec3d([(-1,-2,-3),(-4,-5,-6)]) # commas required
test.value =   [(0,1,2),(3,4,5)]  # commas required
# test = MFVec3d( 1,2,3,4,5,6 ) # TODO
print ("MFVec3d test       =", test)
print ("MFVec3d test.value =", test.value)
# - - - - - - -
test = SFVec4f()
test = SFVec4f((-1,-2,-3,-4)) # commas required
test.value = (1, 2, 3, 4)     # commas required
print ("SFVec4f test       =", test)
print ("SFVec4f test.value =", test.value)

test = MFVec4f()
test = MFVec4f([(-1,-2,-3,-4),(-5,-6,-7,-8)]) # commas required
test.value =   [(0,1,2,3),(4,5,6,7)]  # commas required
# test = MFVec4f( 1,2,3,4,5,6,7,8 ) # TODO
print ("MFVec4f test       =", test)
print ("MFVec4f test.value =", test.value)

test = SFVec4d()
test = SFVec4d((-1,-2,-3,-4)) # commas required
test.value = (1, 2, 3, 4)     # commas required
print ("SFVec4d test       =", test)
print ("SFVec4d test.value =", test.value)

test = MFVec4d()
test = MFVec4d([(-1,-2,-3,-4),(-5,-6,-7,-8)]) # commas required
test.value =   [(0,1,2,3),(4,5,6,7)]  # commas required
# test = MFVec4d( 1,2,3,4,5,6,7,8 ) # TODO
print ("MFVec4d test       =", test)
print ("MFVec4d test.value =", test.value)
# - - - - - - -
test = SFColor()
```

```
#test.value = (0, .5, 1, 5) # 4 elements, illegal tupleSize
#test.value = (0, .5, 5)    # illegal value 5
test = SFColor((0, .5, 1)) # commas required
test.value = (0, .5, 1)    # commas required
print ("SFColor test       =", test)
print ("SFColor test.value =", test.value)
# - - - - - - - -
test = MFColor()
test = MFColor([(0, .5, 1),(1, .5, 0)]) # commas required
test.value =   [(0, .5, 1),(1, .5, 0)]  # commas required
# test.value = (0, .5, 1, 5)    # illegal value 5
# test = MFColor([0, .5, 1, 1, .5, 0]) # TODO
print ("MFColor test       =", test)
print ("MFColor test.value =", test.value)
# - - - - - - - -
test = SFColorRGBA()
test = SFColorRGBA((0, .5, 1, 0.75)) # commas required
test.value = (0, .5, 1, 0.75)    # commas required
print ("SFColorRGBA test       =", test)
print ("SFColorRGBA test.value =", test.value)

test = MFColorRGBA()
test = MFColorRGBA([(0, .5, 1, 0.75),(1, .5, 0, 0.75)]) # commas required
test.value =   [(0, .5, 1, 0.75),(1, .5, 0, 0.75)]  # commas required
# test = MFColorRGBA( 0, .5, 1, 0.75, 1, .5, 0, 0.75 ) # TODO
print ("MFColorRGBA test       =", test)
print ("MFColorRGBA test.value =", test.value)
# - - - - - - - -
test = SFRotation()
test = SFRotation((0, .5, 1, 0.75)) # commas required
test.value = (0, .5, 1, 0.75)    # commas required
print ("SFRotation test       =", test)
print ("SFRotation test.value =", test.value)

test = MFRotation()
test = MFRotation([(0, .5, 1, 0.75),(1, .5, 0, 0.75)]) # commas required
test.value =   [(0, .5, 1, 0.75),(1, .5, 0, 0.75)]  # commas required
# test = MFRotation( 0, .5, 1, 0.75, 1, .5, 0, 0.75 ) # TODO
print ("MFRotation test       =", test)
print ("MFRotation test.value =", test.value)
# - - - - - - - -

test = SFNode()
test = SFNode(WorldInfo(DEF='A'))
print ("SFNode() test       =", test)
print ("SFNode() test.value =", test.value)

test = MFNode([Group(DEF='B'),WorldInfo(DEF='C')])
print ("MFNode()     test             =",     test)
print ("MFNode()     test.getValue() =",     test.getValue(), '(utility method)')
print ("MFNode() str(test.value)      =", str(test.value), 'TODO get result to match, avoid need for getValue()')

# - - - - - - - -
materialInstance = Material()
materialInstance = Material(diffuseColor=(0.5,0.5,0.5), transparency=0.2, DEF='Grey')
print('materialInstance.__name__=', materialInstance.__name__)

print("field accessor test, including default value emissiveColor:")
print("materialInstance=" + materialInstance.__name__ +
    "(DEF='" + str(materialInstance.DEF) +
    "',diffuseColor=" + str(materialInstance.diffuseColor) +
    ",emissiveColor=" + str(materialInstance.emissiveColor) + # exposes default value
    ",transparency=" + str(materialInstance.transparency) + ")")
print('must use str() function when concatenating:')
print('    materialInstance  =',       materialInstance)
print('str(materialInstance) = ' + str(materialInstance) + ' (should match)')

print('isValidSFNode (materialInstance) =' + str(isValidSFNode (materialInstance)))
print('isX3DNode     (materialInstance) =' + str(isX3DNode     (materialInstance)))
print('isX3DStatement(materialInstance) =' + str(isX3DStatement(materialInstance)))

# print('type(materialInstance) =',type(materialInstance))

# import inspect
# from inspect import signature
# print(inspect.getmembers(str))

print("WorldInfo(USE='useful',class_='classic')=",WorldInfo(USE='useful',class_='classic'))
print("    Group() =",    Group() )
print("str(Group())=",str(Group()) + ' (should match)')

routeInstance = ROUTE(fromField="Here",toField="There")
print('    routeInstance =',     routeInstance)
print('str(routeInstance)=', str(routeInstance) + ' (should match)')

print('    ROUTE()  =',      ROUTE())
print('str(ROUTE()) =',  str(ROUTE()) + ' (should match)') # must use str() function when concatenating in print statement

print('isX3DNode     (routeInstance)=' + str(isX3DNode     (routeInstance)))
print('isX3DNode     (ROUTE())      =' + str(isX3DNode     (ROUTE())) + ' (should match)')
print('isX3DStatement(routeInstance)=' + str(isX3DStatement(routeInstance)))
print('isX3DStatement(ROUTE())      =' + str(isX3DStatement(ROUTE())) + ' (should match)')

nestedNodesTest = Shape(
    appearance=Appearance(
        material=Material(diffuseColor=(0.5,0.5,0.5,6), transparency=0.2, DEF='Grey')),
    geometry=Sphere(radius=2),
    metadata=MetadataString(value='checking')) # TODO isValidMFString should fail when not a list
print ('    nestedNodesTest =',     nestedNodesTest)
print ('str(nestedNodesTest)=', str(nestedNodesTest) + ' (should match)')

groupTest = Group(bboxSize=[1,2,3])
groupTest = Group(
   bboxSize=[1,2,3],                          # simple fields and
   children=[WorldInfo(),Group(),Shape()])  # MFNode child list
print ('    groupTest  =',     groupTest)
print ('str(groupTest) =', str(groupTest) + ' (should match)')
# Group(WorldInfo(),bboxSize=[1,2,3]) # possible? maybe not needed

headTest = head()
headTest = head(children=[component(),unit(),meta(name='1',content='2'),meta()])
#headtest.children=[component(),unit(),meta(name='1',content='2'),meta()] # TODO fails
print ('     headTest  =',     headTest )
print (' str(headTest) =', str(headTest) + ' (should match)')

sceneTest = Scene() # children=[WorldInfo(),Group()]
sceneTest = Scene(children=[WorldInfo(),Group()])
#sceneTest.children=[WorldInfo(),Group(),Shape()]
print ('    sceneTest  =',     sceneTest)
print ('str(sceneTest) =', str(sceneTest) + ' (should match)')

modelTest = X3D(
            head=head(
                children=[
                      meta(name="description", content="name-value pair"),
                      meta(name="description2",content="name-value pair2"),
                      meta(name="info",   content="diagnostic test 1"),
                      meta(name="hint",   content="diagnostic test 2"),
                      meta(name="warning",content="diagnostic test 3"),
                      meta(name="error",  content="diagnostic test 4")]
            ),
            Scene=Scene(children=[WorldInfo(),Group()])) # Scene=Scene(children=[WorldInfo(),Group()])  Scene=None
print ('    modelTest  =',     modelTest)
print ('str(modelTest) =', str(modelTest) + ' (should match)')

print ()
print ("metaDiagnostics utility function:")
print ( metaDiagnostics(modelTest))

print ()
print ("TODO value range checks for simple types")
print ("TODO check node types")
print ("TODO add and invoke validation methods that walk model tree")

# TODO requires *arg and node-type-checking support
# Appearance(         Material(diffuseColor=(0.5,0.5,0.5), transparency=0.2, DEF='Grey'))

print ("PythonX3dSmokeTests smoke tests complete.")
```

```
===================
Importing local development copy of X3D package:
  from x3d import *
===================
PythonX3dSmokeTests:
SFBool test       = True
SFBool test.value = True
MFBool test       = [True, False, True, False, True, False]
MFBool test.value = [True, False, True, False, True, False]
SFInt32 test       = 1
SFInt32 test.value = 1
MFInt32 test       = [0, 1, 2, 3, 4, 5]
MFInt32 test.value = [0, 1, 2, 3, 4, 5]
SFFloat test       = 1
SFFloat test.value = 1
MFFloat test       = [0, 1, 2]
MFFloat test.value = [0, 1, 2]
SFDouble test       = 1
SFDouble test.value = 1
MFDouble test       = [0, 1, 2]
MFDouble test.value = [0, 1, 2]
SFString test       = test setter
SFString test.value = test setter
MFString test       = [test, setter]
MFString test.value = ['test', 'setter']
SFVec2f test       = (1, 2)
SFVec2f test.value = (1, 2)
MFVec2f test       = [(0, 1), (2, 3)]
MFVec2f test.value = [(0, 1), (2, 3)]
SFVec2d test       = (1, 2)
SFVec2d test.value = (1, 2)
MFVec2d test       = [(0, 1), (2, 3)]
MFVec2d test.value = [(0, 1), (2, 3)]
SFVec3f test       = (1, 2, 3)
SFVec3f test.value = (1, 2, 3)
MFVec3f test       = [(0, 1, 2), (3, 4, 5)]
MFVec3f test.value = [(0, 1, 2), (3, 4, 5)]
SFVec3d test       = (1, 2, 3)
SFVec3d test.value = (1, 2, 3)
MFVec3d test       = [(0, 1, 2), (3, 4, 5)]
MFVec3d test.value = [(0, 1, 2), (3, 4, 5)]
SFVec4f test       = (1, 2, 3, 4)
SFVec4f test.value = (1, 2, 3, 4)
MFVec4f test       = [(0, 1, 2, 3), (4, 5, 6, 7)]
MFVec4f test.value = [(0, 1, 2, 3), (4, 5, 6, 7)]
SFVec4d test       = (1, 2, 3, 4)
SFVec4d test.value = (1, 2, 3, 4)
MFVec4d test       = [(0, 1, 2, 3), (4, 5, 6, 7)]
MFVec4d test.value = [(0, 1, 2, 3), (4, 5, 6, 7)]
SFColor test       = (0, 0.5, 1)
SFColor test.value = (0, 0.5, 1)
MFColor test       = [(0, 0.5, 1), (1, 0.5, 0)]
MFColor test.value = [(0, 0.5, 1), (1, 0.5, 0)]
SFColorRGBA test       = (0, 0.5, 1, 0.75)
SFColorRGBA test.value = (0, 0.5, 1, 0.75)
MFColorRGBA test       = [(0, 0.5, 1, 0.75), (1, 0.5, 0, 0.75)]
MFColorRGBA test.value = [(0, 0.5, 1, 0.75), (1, 0.5, 0, 0.75)]
SFRotation test       = (0, 0.5, 1, 0.75)
SFRotation test.value = (0, 0.5, 1, 0.75)
MFRotation test       = [(0, 0.5, 1, 0.75), (1, 0.5, 0, 0.75)]
MFRotation test.value = [(0, 0.5, 1, 0.75), (1, 0.5, 0, 0.75)]
```

```
SFNode() test       = WorldInfo(DEF='A')
SFNode() test.value = WorldInfo(DEF='A')
MFNode()     test            = [Group(DEF='B'), WorldInfo(DEF='C')]
MFNode()     test.getValue() = [Group(DEF='B'), WorldInfo(DEF='C')] (utility metho
d)
MFNode() str(test.value)     = [<x3d.Group object at 0x00000294745DFBA8>, <x3d.Wor
ldInfo object at 0x00000294745DF9B0>] TODO get result to match, avoid need for get
Value()
materialInstance.__name__= Material
field accessor test, including default value emissiveColor:
materialInstance=Material(DEF='Grey',diffuseColor=(0.5, 0.5, 0.5),emissiveColor=
(0, 0, 0),transparency=0.2)
must use str() function when concatenating:
    materialInstance  = Material(DEF='Grey',diffuseColor=(0.5, 0.5, 0.5),transpare
ncy=0.2)
str(materialInstance) = Material(DEF='Grey',diffuseColor=(0.5, 0.5, 0.5),transpare
ncy=0.2) (should match)
isValidSFNode (materialInstance) =True
isX3DNode     (materialInstance) =True
isX3DStatement(materialInstance) =False
WorldInfo(USE='useful',class_='classic')= WorldInfo(USE='useful',class_='classic')
    Group() = Group()
str(Group())= Group() (should match)
    routeInstance = ROUTE(fromField='Here',toField='There')
str(routeInstance)= ROUTE(fromField='Here',toField='There') (should match)
    ROUTE()  = ROUTE()
str(ROUTE()) = ROUTE() (should match)
isX3DNode     (routeInstance)=False
isX3DNode     (ROUTE())      =False (should match)
isX3DStatement(routeInstance)=True
isX3DStatement(ROUTE())      =True (should match)
    nestedNodesTest = Shape(appearance=Appearance(material=Material(DEF='Grey',dif
fuseColor=(0.5, 0.5, 0.5, 6),transparency=0.2)),geometry=Sphere(radius=2),IS=Metad
ataString(value='checking'))
str(nestedNodesTest)= Shape(appearance=Appearance(material=Material(DEF='Grey',dif
fuseColor=(0.5, 0.5, 0.5, 6),transparency=0.2)),geometry=Sphere(radius=2),IS=Metad
ataString(value='checking')) (should match)
    groupTest  = Group(bboxSize=[1, 2, 3],children=[WorldInfo(), Group(), Shape
()])
str(groupTest) = Group(bboxSize=[1, 2, 3],children=[WorldInfo(), Group(), Shape
()]) (should match)
     headTest  = head(children=[component(), unit(), meta(content='2',name='1'), m
eta()])
 str(headTest) = head(children=[component(), unit(), meta(content='2',name='1'), m
eta()]) (should match)
    sceneTest  = Scene(children=[WorldInfo(), Group()])
str(sceneTest) = Scene(children=[WorldInfo(), Group()]) (should match)
    modelTest  = X3D(head=head(children=[meta(content='name-value pair',name='desc
ription'), meta(content='name-value pair2',name='description2'), meta(content='dia
gnostic test 1',name='info'), meta(content='diagnostic test 2',name='hint'), meta
(content='diagnostic test 3',name='warning'), meta(content='diagnostic test 4',nam
e='error')]),Scene=Scene(children=[WorldInfo(), Group()]))
str(modelTest) = X3D(head=head(children=[meta(content='name-value pair',name='desc
ription'), meta(content='name-value pair2',name='description2'), meta(content='dia
gnostic test 1',name='info'), meta(content='diagnostic test 2',name='hint'), meta
(content='diagnostic test 3',name='warning'), meta(content='diagnostic test 4',nam
e='error')]),Scene=Scene(children=[WorldInfo(), Group()])) (should match)

metaDiagnostics utility function:
meta info: diagnostic test 1, hint: diagnostic test 2, warning: diagnostic test 3,
error: diagnostic test 4
```

```
TODO value range checks for simple types
TODO check node types
TODO add and invoke validation methods that walk model tree
PythonX3dSmokeTests smoke tests complete.
```